Universidad Autonoma de Madrid

Escuela Politécnica Superior

Trabajo Fin de Máster

SmartBus: Big Data & Data Science en Transporte Urbano

Máster Universitario en Ingeniería Informática

Autor: Carlos Rosado Moral

Tutor: Manuel Sánchez-Montañés Isla

Septiembre, 2017

SmartBus: Big Data & Data Science en Transporte Urbano

AUTOR: Carlos Rosado Moral

TUTOR: Manuel Sánchez-Montañés Isla

Dpto.Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Septiembre 2017

«*Duda siempre de ti mismo, hasta que los datos no dejen lugar a dudas.*»

Louis Pasteur, 1822-1895

Agradecimientos

Una de las partes más significativas de un proyecto es sin ningún lugar a duda los agradecimientos a aquellas personas que han hecho posible que este proyecto se haya realizado y haya seguido adelante.
Las personas más importantes que me han ayudado tanto al finalizar los estudios como durante estos han sido sobre todo mi familia. Por ello quiero dar las gracias a mis padres y a toda mi familia en particular, por haber sido constantes en cuanto a su apoyo aportado, al igual que yo se propusieron una meta, la cual era que siempre fuese bueno en lo que hacía, que fuese constante y que ante todo hiciese algo con lo que me sintiese cómodo, una vez más muchas gracias por todo.

Por supuesto no quiero olvidarme de mis tutores, en primer lugar, dar las gracias a Manuel Sánchez-Montañes, porque siempre ha estado dispuesto a resolverme todas las dudas que se han mostrado en este proyecto y porque siempre confió en mi cuando le propuse este fabuloso tema del cual no me arrepiento. En segundo lugar, quiero hacer una especial mención a los tutores y gente que me ha ayudado desde mi empresa (Synergic Partners, Telefónica Company).

Por último dar las gracias a la EMT que siempre ha estado dispuesta a ayudar y a enviarnos datos en cualquier momento y situación, y que sin ellos el proyecto no se hubiese podido realizar.

Resumen

El Trabajo de Fin de Máster (TFM) que se presenta trata sobre la realización de un estudio y análisis de ciertos tipos de datos procedentes de la red de autobuses de Madrid (EMT). El sistema se basa en una plataforma Big Data mediante la cual se almacenarán grandes cantidades de datos procedentes de diversas fuentes, tanto internas a la EMT como externas, con el fin de aplicar técnicas de aprendizaje automático para segmentar las líneas de autobús y posteriormente realizar una predicción de la demanda diaria en una de ellas. Se realizará un análisis de los resultados obtenidos mediante el cual se pueda mejorar la eficiencia de la red de autobuses de la EMT por medio del análisis de técnicas de regresión y clustering en grandes cantidades de datos. Para finalizar, se implementará un DashBoard mediante el cual el usuario final pueda ver los resultados de nuestro estudio y pueda tomar decisiones en base a un criterio basado en grandes cantidades de datos.

Palabras clave: Big Data, Machine Learning, Smart City, Hadoop, Spark, Hive, Deep Learning, Regresión.

Abstract

In this Final Master's Project (TFM) we study and analyse real data from the Madrid bus network (EMT). The system presented here is a Big Data platform by which large amounts of data from various sources, both internal to the EMT and external, are stored. Then machine learning techniques are applied to obtain a segmentation of the bus lines and to construct of a predictive model for the daily number of travelers that will use a particular bus line. We will perform an analysis of the results that can be used to improve the efficiency of the EMT bus network. In this analysis we will use regression techniques and clustering in large amounts of data. Finally, a DashBoard will be implemented in which the end user can see the results of our study and make decisions based on a criterion supported by large amounts of data.

Keywords: Big Data, Machine Learning, Smart City, Hadoop, Spark, Hive, Deep Learning, Regression

Índice de Contenido

Índice de Figuras

Índice de Tablas

Glosario de Términos

Término	Descripción
Hadoop	Apache Hadoop es un framework de software que soporta aplicaciones distribuidas bajo una licencia libre. Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos. Hadoop se inspiró en los sistemas de Google para MapReduce y Google File System (GFS).
Hive	Apache Hive es una infraestructura de data warehouse construida sobre Hadoop para proporcionar sumarización, consulta y análisis de datos. Hive proporciona una interfaz similar a SQL para consultar los datos almacenados en diversas bases de datos y sistemas de archivos que se integran con Hadoop.
Spark	Spark es un framework Open-Source de computación en cluster. Originalmente desarrollado en la Universidad de California, AMPLab Berkeley, la base de Spark fue más tarde donado a la Apache Software Foundation, que lo ha mantenido desde entonces. Spark proporciona una interfaz para programar clusters completos con paralelismo de datos y tolerancia a fallos.
Smart City	Smart City o Ciudad Inteligente, se refiere a un tipo de desarrollo basado en la sostenibilidad que es capaz de responder adecuadamente a las necesidades básicas de las instituciones, empresas, y de los propios habitantes, tanto en el plano económico, como en los aspectos operativos, sociales y ambientales.
Machine Learning	El Machine Learning o Aprendizaje Automático es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan a los sistemas y ordenadores aprender, es decir mediante el Machine Learning podemos crear programas capaces de generalizar comportamientos a partir de una información suministrada a partir de ejemplos o muestras.

Regresión	Proceso estadístico para estimar el valor de una o varias variables numéricas llamadas dependientes, en función de otras variables llamadas variables independientes o predictoras.
Cloudera	Cloudera es una compañía que proporciona software basado en Apache Hadoop. La distribución Open Source de Apache Hadoop, CDH (Cloudera Distribution Hadoop) se enfoca en el desarrollo de esta tecnología para empresas. Otro sistema clave de Cloudera es el Cloudera Manager, mediante el cual el usuario puede monitorizar y controlar todo el cluster Cloudera.
QGIS	Es un sistema de Información Geográfica(SIG) de código libre para diversas plataformas. Permite manejar formatos raster y vectoriales a través de las bibliotecas GDAL y OGR, así como bases de datos. Una de las grandes versatilidades de QGIS es su facilidad de interconexión con muchas bases de datos geoespaciales.
SAX	SAX (Symbolic Aggregate Approximation) es una técnica para representar y reducir la dimensionalidad en series temporales. Su funcionamiento se basa en dos fases: (1) Realización de un PAA (Piecewise Aggregate Approximation, y (2) conversión de la secuencia de PAA en letras para la reducción de la serie.
ARIMA	ARIMA (autoregressive integrated moving average) es un tipo de modelo autorregresivo integrado de media móvil que utiliza regresiones estadísticas con el fin de encontrar patrones para realizar una predicción de los futuros datos.
Deep Learning	Deep Learning es un tipo de algoritmos de aprendizaje automático basados en redes neuronales de múltiples capas que intentan obtener abstracciones de alto nivel de un conjunto grande de datos. La profundidad de las redes puede ser espacial (como en las redes convolucionales) o temporal (como en las redes LSTM).
LSTM	LSTM es una arquitectura de red neuronal artificial recurrente que permite relacionar eventos separados a una distancia temporal arbitraria.

Keras	Keras es una API de alto nivel enfocada en Deep Learning. Está escrita en Python y se puede ejecutar por encima de TensorFlow, CNTK o Theano.

1. Introducción

Este proyecto se basa en el desarrollo y aplicación de las tecnologías Big Data y Data Science en el área del transporte público. En primer lugar, se ha realizado un análisis de los tipos de transporte público y de sus necesidades para la comunidad de Madrid, tras dicho análisis se procedió a optar por realizar nuestro proyecto para la flota de autobuses de la EMT (Empresa Municipal de Transportes) de Madrid, centrándonos en toda su flota de autobuses incluyendo los autobuses nocturnos. Una vez que se obtuvieron los datos principales de nuestro estudio, se realizó un estudio de las posibles fuentes externas que nutran de información adicional a nuestro estudio.

Seguidamente, se procedió a la ingesta de datos en una plataforma Hadoop dentro del cluster del que disponemos, esta es una parte fundamental en todo proyecto de Big Data, ya que los datos tienen que estar de una forma coherente y correctamente integrados para proceder a su posterior análisis, para ello se ha usado la herramienta Hive mediante la cual se han insertado los datos procedentes de csv y bases de datos a nuestra base de datos en Hadoop.

Una vez tenemos los datos insertados en nuestra base de datos, se procedió a realizar una auditoría con el objetivo de detectar incongruencias y posibles aplicaciones en nuestro estudio. A partir de esta fase del proyecto, se empleo la herramienta Spark mediante la cual podemos trabajar con los datos insertados en una base de datos Hadoop de una forma paralela.

Por otro lado, completadas las fases de trabajo con la base de datos y auditoría de datos, procedemos a la segunda parte más interesante de nuestro proyecto, mediante la cual vamos a aplicar técnicas de Data Science y Machine Learning para la predicción de la demanda de viajeros en cada línea de autobuses. Para ello nos apoyaremos en técnicas de regresión que veremos a lo largo de la presente memoria. Este estudio servirá a la EMT para mejorar en costes su red de servicios de autobuses, incrementando o disminuyendo la frecuencia de sus autobuses basados en la predicción de viajeros. Hay que indicar que se verán mas aplicaciones y posibles usos de estas tecnologías que ayuden a la EMT a mejorar su servicio.

Finalmente, vamos a realizar un sistema de visualización mediante una herramienta basada en un dashboard que permita a la EMT hacer uso de nuestro sistema, así como representar la base de datos de una forma más intuitiva y visual, de tal forma que se pueda tomar decisiones basadas en los datos.

1.1. Motivación

El objetivo global es llevar a cabo un trabajo de campo en el área del Big Data y Data Science. En los últimos años se está hablando mucho de las Smart Cities y de cómo estas cambiarán la forma de operar de los ciudadanos, y en este área se han realizado grandes proyectos como el de la ciudad de Songdo (Corea del Sur) la cual es un referente en este campo, ya que tiene una gran cantidad de sistemas Big Data orientados a las Smart Cities.

De este modo, hemos detectado una gran necesidad en Madrid para llevar las tecnologías Big Data al ámbito de la ciudad, por lo que se realizó un análisis de las posibles aplicaciones de esta tecnología. En primer lugar, se vieron las pocas fuentes de datos en este ámbito, ya que se empezó a realizar un análisis de la operativa del metro de Madrid llegando a la conclusión de que los datos guardados no eran lo suficientemente buenos como para realizar nuestro estudio. Por lo que se optó por en enfocarnos en analizar los datos del otro medio de transporte más importante en la ciudad, los autobuses de la EMT. Hay que indicar que estos autobuses son de los únicos transportes que se mueven por casi todas las calles de la ciudad casi y a todas las horas del día, lo que nos proporciona una visión más amplia del comportamiento del tráfico y de las personas en la ciudad.

Para cualquier empresa de transporte, ya sea de personas como de objetos, es importante el análisis de las rutas de sus autobuses o camiones, ya que una mejora en cualquier punto de estas puede ser una mejora en la calidad del servicio o una mejora económica. Con nuestro estudio se pretende dar una mejora en la calidad del servicio de la EMT que implique que sus usuarios estén más satisfechos con el servicio ofrecido, así como una optimización de los recursos de la propia compañía.

1.2. Objetivos

A continuación se expone una lista de objetivos concretos que se pretenden satisfacer con la elaboración de este trabajo, así como las áreas en las que se engloban cada uno de ellos:

- **Objetivo 1: Estudio de los datos de transporte público**
  Se realizará un estudio de los datos que proceden del transporte público, realizando una investigación en dicho campo para entender los problemas a resolver, así como los posibles comportamientos de los datos en este entorno.

- **Objetivo 2: Estudio de fuentes de datos externas al transporte público**
  Se realizará un estudio de los posibles datos que afectan a los datos del transporte público (meteorología, tráfico, eventos, etc) y que por lo tanto se tienen que tener en cuenta para nuestro análisis.

- **Objetivo 3: Estudio de la ingesta de datos en Hadoop**
  Se realizará un estudio de los dos tipos de ingesta que se tienen que tener en cuenta en este tipo de problemas:

  - **Datos históricos:** Ingesta de los datos proporcionados por la EMT y las fuentes externas, de años y meses anteriores que sirvan como entrenamiento para la posterior predicción.
  - **Datos Actuales:** ingesta de los datos actuales sobre los que se realizará la predicción teniendo en cuenta los datos futuros.

- **Objetivo 4: Análisis de los datos en el transporte público.**
  Tras realizar el estudio de los datos del sector y la ingesta de nuestros datos, el siguiente objetivo a realizar es realizar un correcto análisis de las bases de datos de las que disponemos, de tal forma que nos ayude a la hora de generar un buen modelo. Para esta fase se ha realizado un trabajo de investigación en el campo de la auditoría de datos.

- **Objetivo 5: Técnicas de segmentación y predicción de la demanda.**
  El objetivo principal de nuestro problema es la segmentación y predicción de la demanda en el transporte público. Por lo tanto se realizará un estudio en dicho campo que nos permita entender las principales técnicas y algoritmos usados en este área.

- **Objetivo 6: Visualización de los resultados.**
  Finalmente, se procederá a realizar un estudio sobre el uso de técnicas de visualización para darle al usuario una mayor comodidad a la hora de interpretar los datos, así como de las predicciones utilizadas. El objetivo es que el usuario sea capaz de tomar decisiones de una forma más coherente.

1.3. Estructura del Documento

Este documento se encuentra definido en siete partes claramente diferenciadas:

- **Capítulo I y II:** En estos capítulos se encuentra la introducción de nuestro trabajo y el estado del arte.

- **Capítulos III y IV:** En estos capítulos vamos a ver las fases de ingesta y preprocesamiento de datos para su correcto tratamiento y uso.

- **Capítulo V:** Esta es una de las fases más importantes en nuestro trabajo, ya que es donde vamos a aplicar diferentes tipos de modelos predictivos.
- **Capítulo VI:** En esta fase vamos a centrarnos en la visualización orientada en el usuario final.
- **Capítulo VII:** Finalmente exponemos las conclusiones y trabajo futuro.

1.4. Material Empleado

Para la realización de nuestro proyecto se ha usado un cluster cedido por la empresa en la cual trabaja el autor del presente trabajo. Este cluster tiene las siguientes características:

- **Distribución Cloudera:** esta es una distribución usada para proyectos Big Data la cual tiene instaladas otras distribuciones y programas como Hadoop, Spark, etc.
- **Número de nodos:** nuestro cluster trabaja con 4 nodos activos.
- **Tamaño de la RAM:** el tamaño de la memoria RAM es de 160Gb que se distribuye en todos los nodos que hemos mencionado.
- **Número de Cores:** el número de cores que disponemos es de 56.

Hay que indicar que este no es un cluster relativamente grande para proyectos Big Data, pero para nuestro proyecto es aplicable dado el volumen de nuestros datos.

2. Estado del Arte

Este capítulo introducirá diversos aspectos relacionados con los estudios realizados para abordar nuestro proyecto, así como los tipos de algoritmos que vamos a utilizar. En la sección 2.1 se hará un repaso del estado del arte de predicción de la demanda en sistemas similares al tratado. A continuación, en la sección 2.2 y 2.3 se realiza una breve explicación los métodos de clasificación no supervisada y regresión utilizados en este trabajo. Nuestro objetivo es realizar una segmentación de las lineas de la EMT y realizar una predicción de la demanda de estas lineas. Finalmente se hará una introducción a las redes neuronales LSTM empleadas mediante Deep Learning para realizar una predicción de una serie temporal.

2.1. Revisión de trabajos anteriores

Como vimos en el capítulo anterior, hay ciudades que ya emplean técnicas de big data y data science en el transporte urbano como por ejemplo la ciudad de Songdo en Corea del Sur, en donde todos los sistemas de transporte público están conectados. Uno de los trabajos referentes en este sector es el libro "The real-time city? Big data and smart urbanism"[**1**], en donde se citan diversos estudios sobre cómo puede afectar un posible uso del Big Data en el transporte y urbanismo en una ciudad.

En cuanto a las técnicas de machine learning utilizadas para el transporte público hay que indicar que la gran mayoría de estudios hacen referencia a los algoritmos genéticos empleados para la mejora en las rutas de los autobuses, metros, etc. [**2**], por lo que este es un tema muy común en este sector, ya que todas las ciudades desean optimizar sus rutas para mejorar la calidad del servicio ofrecido a los ciudadanos. Adicionalmente, hemos encontrado una serie de artículos ([**3**], [**4**] y [**5**]) los cuales hablan de predicciones en series temporales en el ámbito del transporte público. De ellos se han podido sacar ideas de los algoritmos y fuentes externas que necesitábamos emplear para construir nuestro modelo, aunque dichos artículos trataban temas diferentes a lo que nuestro estudio plantea ya que se centraban en la predicción del tráfico, gps, etc. y no en la demanda de los viajeros, que es lo que nos interesa predecir en nuestro trabajo.

Como podemos ver se trata de un tema novedoso dentro del actual mundo del Big Data y Data Science y que por tanto es un campo en pleno desarrollo que no dudamos que irá cada vez a más.

2.2. Algoritmos de clustering

Los algoritmos de clustering (o de clasificación no supervisada) son métodos en donde se intenta separar (o "segmentar") las diferentes observaciones en grupos de tal manera que las observaciones pertenecientes al mismo grupo son muy similares entre sí pero a la vez muy diferentes a las de otros grupos [**18**]. De esta forma, los métodos de clustering intentan detectar la estructura natural del problema. Estos métodos se distinguen de los algoritmos supervisados en que no hay ningún tipo de referencia a priori, es decir no tenemos una clase a predecir o variable predictora. De esta forma los algoritmos de aprendizaje no supervisados tratan los conjuntos de datos de entrada como un conjunto de variables aleatorias, construyendo un modelo para dicho conjunto de datos.

2.2.1. K-Means

El algoritmo K-means es un método de clustering cuyo objetivo es la agrupación de un conjunto de n observaciones en k clusters en el que cada observación pertenece al cluster en el que su valor medio es el más cercano [**17**]. Dicho valor es conocido como centroide de dicho cluster k. Por otra parte, K-means tiende a encontrar clusters con una extensión espacial comparable.

En cuanto a la descripción matemática del algoritmo podemos decir que dado un conjunto de observaciones $(x_1, x_2, ..., x_n)$, donde cada observación es un vector real de d dimensiones, k-means construye una partición de dichas observaciones en k clusters ($k \leq n$) con el fin de minimizar la suma de los cuadrados dentro de cada cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

2.2.2. Clustering Jerárquico

En machine learning, un clustering o agrupamiento jerárquico es un método que busca crear una jerarquía o árbol de clusters [**18**]. Existen principalmente dos técnicas que son muy utilizadas:

- **Aglomerativas:** se trata de un agrupamiento ascendente iterativo: se comienza con tantos clusters como observaciones (cada cluster corresponde a una única observación). En cada iteración siguiente el algoritmo une dos de los clusters existentes (los dos clusters que más se parecen entre sí) formando un cluster mayor.
- **Divisivas:** se trata de un agrupamiento descendente: se comienza con un sólo cluster, al que pertenecen todas las observaciones, y en cada iteración se va dividiendo uno de los clusters existentes en dos.

El resultado de las observaciones suele ser presentado en un dendrograma, que es un tipo de gráfico que representa la jerarquía de clusters construida y que se puede interpretar fácilmente.

Hay que indicar que la elección de una métrica adecuada influenciará en la forma en la que se realizan los grupos de observaciones, ya que algunas observaciones o grupos pueden estar cerca de otros de acuerdo a una distancia y más lejos de acuerdo a otra distancia. En la siguiente tabla podemos ver algunas de las métricas más empleadas para los algoritmos de clustering jerárquico:

Distancia	Formula
Distancia Euclídea	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Distancia Euclídea Cuadrado	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Distancia Manhattan	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
Distancia Máxima	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Distancia Mahalanobis	$\sqrt{(a-b)^\top S^{-1}(a-b)}$
Similitud coseno	$\frac{a \cdot b}{\|a\|\|b\|}$

Tabla 2-1.: Distancias más utilizadas en clustering jerárquico

Otro factor importante a tener en cuenta es el criterio de enlace, el cual determina la distancia entre los conjuntos de las observaciones como una función de las distancias entre las observaciones dos a dos.

2.3. Algoritmos de Regresión

Los algoritmos de regresión son un conjunto de técnicas que intentan predecir, a partir de un conjunto de variables (también llamadas predictores, variables independientes, variables regresoras, o variables explicativas) el valor numérico de otra variable o conjunto de variables (llamadas variables dependientes) [**19**]. En primer lugar, tenemos una matriz de datos expresada exactamente igual que en cualquier tipo de algoritmo de clasificación. Es decir, dado un conjunto de muestras M, donde cada muestra tiene f variables. Se representa como una matriz X en el que cada fila representa un ejemplo y cada columna es una variable.

Además, en el caso de la regresión, denotamos como X_i para i = 1,..., n a la variable aleatoria asociada a la n-ésima variable de la muestra. Por lo tanto, la predicción se realiza sobre un valor continuo Y. Este valor Y continuo recibe el nombre de variable explicada o dependiente, mientras que cada una de las variables de los ejemplos X_1,...,X_n se llaman variables explicativas o regresoras.

2.3.1. Regresión Lineal

En la regresión lineal se asume que la variable que se quiere predecir, y, es una combinación lineal de las variables independientes x_i más un ruido n [**19**]:

$$y = w_0 + \sum_{i=1}^{d} w_i \cdot x_i$$

Los parámetros del modelo (es decir, w_0 y las w_i) se eligen de tal forma que se minimiza el error cuadrático medio del modelo a lo largo del conjunto de observaciones del conjunto de entrenamiento. Dada la sencillez del modelo los valores óptimos de los parámetros se pueden calcular mediante una solución analítica cerrada.

2.3.2. Random Forest Regressor

Como hemos visto, en los problemas de regresión se utiliza todo el espacio de características, pero en problemas con múltiples variables que interactúen de una forma no lineal, si construimos un modelo lineal podemos tener un error muy grande. Los árboles de regresión [**20**] son un método para construir de manera sencilla un modelo no lineal. La idea central es dividir el espacio de características en particiones disjuntas para construir un modelo lineal en cada partición. La idea es similar a la usada en los árboles de decisión, en donde se va construyendo un árbol subdividiendo una y otra vez el espacio de características con divisiones parlelas a los ejes. La única diferencia recae en que en cada hoja se ajusta un modelo lineal usando sólo para los ejemplos que caen en dicha hoja.

Por otra parte el método de Random Forest consiste en construir un conjunto de árboles donde los nodos intermedios han sido elegidos al azar (el atributo por el que se pregunta y el punto de corte) [**21**]. A la hora de realizar una predicción para un dato de entrada determinado, se calculan las predicciones individuales que realiza cada uno de los árboles construidos, y se da como predicción el promedio de dichas predicciones. El método de Random Forest ha demostrado ser muy potente y robusto frente al sobreajuste en multitud de aplicaciones.

2.4. Deep Learning

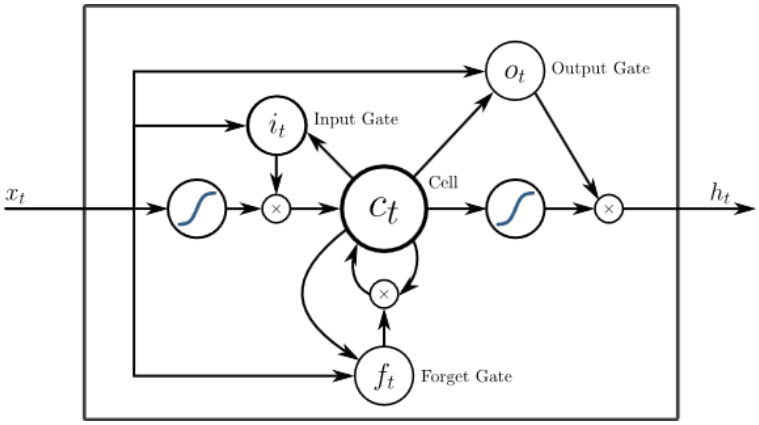
El Deep Learning o Aprendizaje Profundo [**22**, **23**, **24**] es un conjunto de algoritmos de Machine Learning basados en redes neuronales con gran número de capas. Dichos algoritmos intentan modelar abstracciones a alto nivel mediante el uso de arquitecturas compuestas de múltiples transformaciones no lineales. Dichos algoritmos han demostrado un poder de predicción muy bueno en diversos campos tales como el reconocimiento facial, reconocimiento del habla, análisis de lenguaje natural, diagnóstico de enfermedades a partir de imágenes médicas, etc. En nuestro proyecto nos centraremos únicamente en las redes neuronales profundas LSTM, empleadas para la predicción de series temporales.

2.4.1. Redes LSTM

Las redes de neuronas LSTM (Long Short-Term Memory, "memoria a corto-largo plazo")[**25**, **26**] son un tipo de redes neuronales recurrentes donde cada neurona recibe feedback de las demás y aparte tiene un estado interno cuyo valor se modifica de acuerdo a la experiencia: cada neurona aprende a detectar qué patrones concretos activan su memoria particular, cuáles la resetean, y qué información memorizar. De esta forma una neurona particular puede decidir mantener la información sobre un patrón que detectó hace tiempo, hasta que otro patrón concreto actualiza dicha información. La riqueza de dinámicas que puede detectar una red de este tipo es pues enorme, y además se elimina el efecto del "vanishing gradient"que tienen las redes neuronales recurrentes tradicionales (imposibilidad de mantener en memoria información más allá de cierta ventana temporal).
Mediante una red LSTM podemos clasificar y predecir series temporales que presenten dinámicas rápidas mezcladas con dinámicas lentas y cuyas escalas de tiempo desconocemos a priori.

El típico diseño de arquitectura de una LSTM se basa en una arquitectura hardware en paralelo para mejorar la eficiencia y rapidez en el aprendizaje. Los bloques de dicha red contienen varias puertas que controlan el flujo de la información, de tal forma que cuando la puerta tiene un valor cercano a 0 la información no fluye a través de ella, y cuando tienen un valor cercano a 1 fluye. El valor en cada momento de estas puertas depende del estado de las neuronas, el input, y los parámetros W y U que fijan su dinámica, los cuales se van ajustando en el entrenamiento de la red.

En la siguiente imagen podemos ver dicha arquitectura, en donde podemos ver las puertas anteriormente mencionadas:



$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Figura 2-1.: Arquitectura LSTM. Imagen extraída de https://en.wikipedia.org/wiki/Long_short-term_memory

En la imagen anterior se puede observar las diferentes puertas que tiene internamente la arquitectura LSTM. Entre ellas hay una puerta dedicada al olvido (Forget Gate) que controla el grado en el que un valor permanece en la memoria de la neurona. Los parámetros que fijan la dinámica de cada puerta son las matrices W y U correspondientes.

3. Ingesta de Datos

En este capítulo vamos a ver la parte de ingesta de datos de todos los datos provenientes de la EMT (históricos y datos actuales), que denominaremos "fuentes internas", y de todos los datos que no proceden de la EMT, que denominaremos "fuentes externas". Respecto a las fuentes externas estudiaremos qué tipo de fuentes se pueden utilizar en nuestro sistema, así como su tratamiento e integración con las fuentes internas.

3.1. Carga de Datos

En primer lugar, tenemos que realizar la carga de los datos en formato *csv* u otros formatos al sistema HDFS para posteriormente ser insertados en nuestra base de datos. Para ello se han desarrollado dos modelos de script en lenguaje bash que se encargan de cargar los datos desde el propio cluster al sistema HDFS. Estos scripts han sido divididos según la procedencia de sus datos, de tal forma que separemos los datos de fuentes internas de las fuentes externas. Otra forma de subir los datos más efectiva es realizarla desde el propio PC hacia HDFS directamente empleando un sistema webHDFS. En este TFM hemos utilizado ambos sistemas.

Hay que indicar que para la carga normalmente se suele usar Sqoop [**16**] que es un sistema que conecta la base de datos principal (normalmente en SQL) con un cluster Hadoop e inserta automáticamente los datos en sus respectivas tablas siguiendo los criterios previamente indicados en su configuración. Este último método no lo hemos podido realizar ya que no teníamos acceso a los sistemas de la EMT.

Adicionalmente se ha desarrollado un fichero de carga de nuevos datos que es ejecutado mes a mes para realizar las ingestas de datos nuevos.

En el Anexo I podemos encontrar la explicación de dichos scripts así como su enlace a GitHub.

3.2. Ingesta de Datos: Fuentes Internas

En esta sección vamos a ver la ingesta de los datos principales de nuestro estudio, los cuales han sido proporcionados por la EMT.

3.2.1. Históricos de Datos

Hay que diferenciar dos tipos de datos en lo referente al histórico de los datos. En primer lugar vamos a ver los datos de los viajeros y seguidamente veremos los datos referentes a las líneas y paradas.

Datos de Viajeros

El histórico de los datos de viajeros ha sido proporcionado por la EMT mediante petición por correo electrónico. La EMT nos ha enviado por Wetransfer los datos de todas las líneas de autobuses por cada mes de los años 2015, 2016 y 2017. Dichos datos se obtienen de las máquinas de los autobuses que registran la entrada de los viajeros. Cada vez que un viajero inserta su ticket o abono en la máquina esta genera una nueva entrada en la base de datos y queda registrado como nuevo viaje. Las variables que contiene cada tabla de datos son:

- **Fecha:** Campo que nos indica la fecha formado por año, mes y día (ej. 20160101)
- **Instante:** Campo que nos indica el instante de tiempo en el que se ha registrado el evento (ej. 18:01)
- **Línea:** Línea donde se ha producido el evento (ej. 1)
- **Parada:** Parada donde se registra el evento (ej. 167)
- **Título:** Nos indica el tipo de abono que tiene el usuario (ej: 113)
- **BUS:** Nos indica el ID del autobús en el cual se está realizando el viaje (ej. 8327)
- **nviaje:** Nos indica el número del viaje que realiza el autobús. Este campo empieza en 1 y se va incrementando mediante el número de viajes que realiza el autobús (ej. 2)
- **Sentido:** Nos indica el sentido (IDA=1/VUELTA=2) del viaje (ej. 2)
- **Viajeros:** Nos indica el número de viajeros que se han montado en dicho instante de tiempo (ej. 3)

Como ya hemos indicado anteriormente, tenemos 26 ficheros CSV (hasta febrero de 2017, adicionalmente tenemos los datos hasta Julio pero cuando nos los facilitaron ya teníamos implementados y entrenados los modelos) con dicha estructura, uno por cada mes de los años 2015, 2016 y 2017, cuyo espacio es de más de 50Gb en texto plano.

Una vez tenemos identificados los datos, procedemos a la ingesta de estos en nuestra plataforma Hadoop. En primer lugar, ya que dichos ficheros no estaban delimitados adecuadamente, procedimos a guardarlos separados por ' ; ' ya que esto nos mejora su ingesta, esto se realizó mediante un pequeño script en Spark que leía los datos y los guardaba con el nuevo formato

(debido a la sencillez de dicho script no se ha considerado relevante incluirlo en nuestra memoria).

El siguiente paso es realizar la ingesta de cada mes en una tabla en Hadoop. Para ello se ha realizado un script que nos inserta dichos datos mediante Hive (1_carga_viajeros.sh que realiza llamadas a scripts de hadoop, en el Anexo I podemos verlo en más detalle). Una vez tenemos cargados todos los meses en dichas tablas procedemos a insertar todos los meses en una nueva tabla por año (viajeros_2015, viajeros_2016 y viajeros_2017) y finalmente juntamos todos ellos en una tabla final la cual indica todos los trayectos del 2015, 2016 y 2017.

Una vez que tenemos los datos en una tabla agrupamos cada año en tramos a partir de los instantes. Se han seleccionado tramos de una hora, ya que veíamos que no era necesario tener en cuenta tramos más pequeños debido a la similitud del comportamiento en una hora.

Seguidamente nos dimos cuenta de que en el campo "sentido"teníamos un 80 % de valores igual a cero. Esto son errores en las máquinas pero para nuestro estudio es muy importante saber el sentido y por lo tanto se preguntó a la EMT cómo podíamos solventar dichos errores. La EMT nos facilitó una solución mediante la cual se podía extrapolar el sentido mediante los campos línea, parada, bus y número de viaje. Una vez se realizó esta operación nos dimos cuenta de que seguían quedando una gran cantidad de ceros y se optó por añadir una solución alternativa que se basaba en obtener el sentido mediante la parada: debido a que el ID de la parada es única (paradas en la misma calle pero en diferentes sentidos tienen IDs diferentes) mediante la ID de la parada podíamos obtener el sentido. Por lo tanto solicitamos a la EMT el fichero que mapea las paradas con el sentido y se realizó un *join* con nuestra tabla. No obstante seguían quedando ceros que al tratarse de errores en la base de datos se optó por eliminarlos. En total se eliminaron unos 43116 registros en las líneas para el 2015 y unas 133814 para el 2016, siendo estos números mucho más pequeños que el número de registros total en la base de datos (más de cinco millones de registros). Por tanto asumimos que el efecto de la eliminación de estos datos de cara a nuestro análisis es muy pequeño. En el Anexo I tenemos el enlace a estos scripts los cuales corresponden a los ficheros 2.1_ingesta_paradas_sentidos.hql y 2.2_join_groupby_sentidos.hql

Por último, se procede a realizar un *group by* en Hive por fecha, línea, sentido y tramo para obtener como resultado una única tabla que nos muestre los datos de 2015,2016 y 2017 por tramo horario. Esto nos deja como resultado un total de 5118909 registros en nuestra base de datos.

Datos de Líneas y Paradas

Otros datos incluidos en las fuentes externas son los datos referentes a las paradas, líneas, tipos de billete, etc. La EMT nos proporcionó una serie de ficheros que son importantes para nuestro análisis:

- **Tipos Billetes:** Se nos proporcionó un csv donde se indican los identificadores de los abonos y su significado.
- **Plazas:** Otro de los ficheros proporcionados ha sido uno que nos indica las plazas que tiene cada uno de los autobuses indicados en la base de datos mediante un identificador.
- **Tipos de Lineas:** Otro fichero que ha sido adaptado previamente por nosotros indica a qué etiqueta hacen referencia los identificadores de las líneas, es decir, qué tipo de línea son. Hemos insertado una columna más en la base de datos para indicar si la línea está dentro de los siguientes grupos definidos: Común, Nocturno, Circular, Microbús, Servicio Especial, Trabajo y Universitario. Los tipos dependen de si son líneas nocturnas, de trabajo, universitarias, etc.

Dichos datos son insertados al igual que los históricos de los datos mediante una serie de scripts que realizan las operaciones necesarias para la ingesta de estos.

3.2.2. Datos Actuales y Futuros

Se ha desarrollado un script para la ingesta de los nuevos datos que nos vaya enviando la EMT. Teniendo en cuenta que estos datos son enviados mediante email y en formato csv, tenemos que realizar la carga de una forma más manual que si realizásemos ingestas de estos desde su base de datos Oracle hacia nuestro sistema Hadoop.

Dicho script de ingesta de nuevos datos lo podemos encontrar en la parte 1_carga\5_carga_nueva.sh y en 2-2_NuevosDatos, mediante el cual se podrán ingestar todos los datos nuevos en las tablas anteriormente creadas y hacer las operaciones necesarias.

3.3. Ingesta de Datos: Fuentes Externas

En esta sección vamos a ver cómo se han tratado las fuentes de datos externas seleccionadas para la mejora de nuestro estudio que han sido proporcionadas por portales OpenData. Se han seleccionado cuatro bases de datos:

- **Calendario:** se ha generado un fichero csv en donde para cada día se ha establecido un campo indicando el día de la semana y otro campo indicando si es festivo o no en la Comunidad de Madrid. Hay que indicar que dicho fichero se ha generado a mano, ya que era más fácil de obtener.

- **AEMET:** otro fichero que es clave para el análisis de nuestro estudio son los datos de las precipitaciones en Madrid. Para ello se ha generado un fichero que contiene todos los datos de las lluvias en la Comunidad de Madrid para los años a estudiar.
- **Tráfico:** también tenemos que tener en cuenta el estado del tráfico para cada día y hora en nuestro análisis. Estos datos los hemos obtenido de la página de OpenData de Madrid.
- **Eventos:** también tenemos que tener en cuenta los eventos de gran importancia que se desarrollan en Madrid, ya que depende de su intensidad estos pueden afectar a los estados de la líneas de la EMT. Hay que indicar que finalmente sólo se han considerado los eventos deportivos de interés masivo como los partidos del Real Madrid y del Atlético de Madrid.

A continuación vamos a ver cómo se han insertado estos datos en nuestra base de datos de Hadoop.

3.3.1. Históricos de Datos

Calendario

En primer lugar hemos accedido a la página http://www.calendarioslaborales.com para obtener el calendario laboral de la Comunidad de Madrid. Utilizando la información de dicha página se han codificado los días festivos como 1 y los laborables como 0. Seguidamente, mediante Excel se han podido generar los días de la semana de los días indicados, así como su correspondiente separación en día, mes y año. Hay que indicar que se podría haber realizado mediante algunas funciones ya implementadas en Python.

Finalmente se ha procedido a la insertar esta información en la base de datos. El script de su ingesta se encuentra en el fichero 3_ingesta_tablas_aux.hql.

AEMET

Los históricos de las precipitaciones se han obtenido de una página que registra todos los históricos de la AEMET (https://datosclima.es/) en cuatro tramos diarios. Se obtuvieron los estos datos diarios de los años 2015, 2016 y 2017 y se preprocesó el dataset para juntar todos los días de un año en un único csv. Tenemos que tener en cuenta cuatro tipos de columnas:

- **Lluvia:** indica si llueve o no en un día.
- **Intensidad:** indica la intensidad de la lluvia en un día.
- **Lluvia por tramo:** tenemos cuatro tramos horarios (0-6, 6-12, 12-18 y 18-24), este campo nos indica si llueve o no en dichos tramos.

- **Intensidad por tramo:** nos indica la intensidad de la lluvia en cada uno de los tramos.

Una vez que tenemos los datos preprocesados, los insertamos en nuestra base de datos (lo podemos ver en el script anteriormente mencionado).

Tráfico

Los históricos de los datos de tráfico se han obtenido desde el OpenData de la comunidad de Madrid. En primer lugar se ha obtenido un fichero con las ubicaciones de los puntos de medida (buscando en la web de OpenData: Tráfico: Intensidad del tráfico, ubicación de los puntos de medida). De todos los ficheros que aparecen en el zip, se ha seleccionado el fichero pmed_trafico.shp, ya que este le podemos cargar en la herramienta de análisis de datos geográficos QGIS. Estos datos representan los puntos de medida del tráfico mediante un polígono representado por una fecha que indica el sentido del tráfico. Por lo tanto procedemos a generar el punto del centroide de la fecha y seguidamente ampliamos este en un radio de 50 metros para indicar que dicho punto de medida registra todo el tráfico en 50 metros a la redonda.

Una vez que tenemos dicho fichero cargado en la herramienta QGIS, procedemos a la carga de los datos de las ubicaciones de las paradas anteriormente visto. Seguidamente procedemos a calcular la intersección entre los puntos de las paradas y los puntos de las medidas. Esto nos indicará el punto de medida por cada parada. Finalmente exportamos dicha intersección en un fichero csv. En la siguiente imagen podemos ver su visualización:

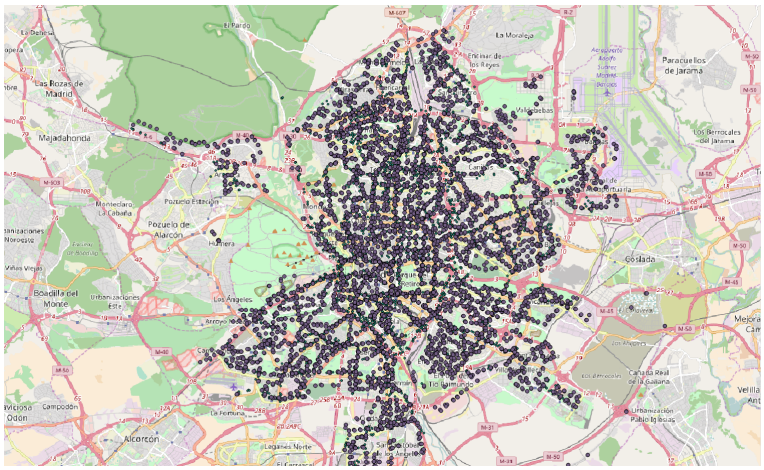


Figura 3-1.: Tráfico por paradas.

Antes de proceder a la carga de dicho fichero en la base de datos nos hemos dado cuenta de que algunas paradas tienen varios puntos de medida del tráfico, por lo que tenemos que

quedarnos con un solo punto. Ya que dichos puntos suelen estar ubicados en la misma calle nos podemos quedar con cualquiera de ellos, por lo que se ha realizado un pequeño script en Python que nos selecciona un único punto de medida para cada parada de cada linea (dado que es un script que solo realiza una eliminación de duplicados, no hemos visto necesario mostrarlo en nuestra memoria). Una vez tenemos dicho fichero, procedemos a su carga en nuestra base de datos.

Finalmente, una vez que tenemos cargados los datos de las paradas con sus puntos de medida del tráfico, procedemos a descargar los datos del tráfico para los años de nuestro estudio, estos han sido previamente preprocesados mediante un pequeño script en Python para que todos los meses tengan el mismo formato. Dichos ficheros los podemos encontrar en la página del OpenData de Madrid (Tráfico: intensidad del tráfico desde julio 2013 (datos de los puntos de medida)). Por último, realizamos un *join* entre dichos datos y los puntos de medida por parada para tener en una tabla las mediciones de tráfico por parada.

Al igual que para los anteriores scripts, en el Anexo I podemos ver el script de esta ingesta en el script 4_ingesta_trafico.hql.

Eventos Deportivos

Dado a que en Madrid uno de los factores que afecta a la demanda de los autobuses son los eventos deportivos y más concretamente los de fútbol, hemos procedido a realizar una ingesta de los datos de eventos de los dos equipos (Real Madrid y Atlético de Madrid) que más pueden provocar una bajada o subida en la demanda de los viajeros. Dado que los datos de los partidos de estos equipos no los hemos encontrado en formato csv, hemos procedido a realizar un scraping de la web (http://www.resultados-futbol.com/) mediante el scraper web import.io con el cual se han obtenido todos los partidos de ambos equipos para los años de nuestro estudio. Las variables de este fichero son:

- **fecha:** esta variable indica la fecha del partido en su formato adecuado.
- **estadioid:** esta variable nos indica el id del estadio (1 para el Santiago Bernabéu y 2 para el Vicente Calderón).
- **equipo_local:** esta variable nos indica el equipo que juega de local dicho partido.
- **equipo_visitante:** esta variable nos indica el equipo que juega de visitante dicho partido.
- **hora:** esta variable nos indica la hora del evento.
- **intensidad:** esta variable nos indica la intensidad del evento mediante los comentarios en la página.

- **tramo_comienzo:** esta variable nos indica dos horas antes de la hora del comienzo del evento.
- **tramo_hora:** esta variable nos indica el curso de las horas del evento.

Una vez que tenemos los datos de estos en un csv, hemos filtrado estos para quedarnos únicamente con los partidos en los que los dos equipos jueguen en casa, ya que pueden afectar bastante al tráfico en Madrid y al comportamiento de algunas líneas de autobuses.

Finalmente se ha creado un fichero csv a mano en donde se establecen las paradas de la EMT más cercanas a los estadios de ambos equipos y que por lo tanto pueden ser afectados por dichos partidos. Para finalizar, se ha realizado un join de las anteriores tablas para obtener las paradas que tienen eventos.

3.3.2. Datos Actuales y Futuros

Hay que indicar que para procesar nuevos datos procedentes de las fuentes externas tenemos que realizar todos los procedimientos anteriormente explicados. Una vez que tengamos los nuevos datasets correspondientes a los nuevos datos a insertar, el siguiente paso es la ejecución del script de carga de nuevos datos anteriormente explicado.

Hay que indicar que esta parte estaría bien tenerla automatizada y poder descargar y procesar los datos automáticamente. No obstante tenemos que indicar que el formato de traspaso de los datos no nos ha permitido realizar esto.

3.4. Join de Fuentes Internas y Externas

Una vez tenemos tanto las fuentes internas como externas insertadas en nuestra base de datos hemos creado una tabla máster que une todas ellas en una única tabla viajeros_sentido_master la cual contiene un total de 609382324 filas.

Finalmente, se ha creado otra tabla máster viajeros_tramos_master la cual realiza un *group by* de la anterior por tramo horario para tener datos de cada una de las líneas agrupadas por su tramo horario. Esta tabla tiene un total de 5118909 registros. Hay que indicar que los tramos horarios seleccionados para las líneas de tipo nocturnas son 23-01,01-03,03-05 y de 05-07, y para el resto de las líneas el tramo horario sería 0-6, 6-12, 12-18 y de 18-24.

Al igual que para las anteriores tablas podemos ver los scripts de las tablas master en el Anexo I y en el script 6_generate_master.hql.

4. Auditoría y preprocesamiento de datos

Una parte importante antes de realizar nuestro estudio predictivo es la fase de auditoría y procesamiento de los datos ya que esto mejora la calidad y robustez del análisis posterior. En los siguientes apartados vamos a explicar las tareas más importantes de todas las realizadas que vemos interesantes para nuestro estudio.
Los scripts para esta parte los podemos encontrar en el Anexo I y en las carpetas 2_IngestaProcesado y 3_Auditoria del repositorio de GitHub.

4.1. Estudio Previo

En primer lugar, hemos considerado relevante, antes de realizar un estudio más profundo de cada una de las líneas, representarlas en un gráfico para chequear su comportamiento. Para ello hemos creado un script que representa el comportamiento de una determinada linea. Dicho script lo podemos ver en el Anexo I en la parte de auditoría en el script 1_Auditoria_EstudioLineas.ipynb.

Tras realizar esto nos dimos cuenta de los siguientes puntos:

- **Laborables vs. Festivos:** el comportamiento de los días laborables (de lunes a viernes) es muy diferente al comportamiento de los fines de semana y festivos. En la siguientes imágenes podemos ver el gráfico de la línea 1 general y de esta misma separada en dos gráficas (una para los días laborables y otra para los días festivos):

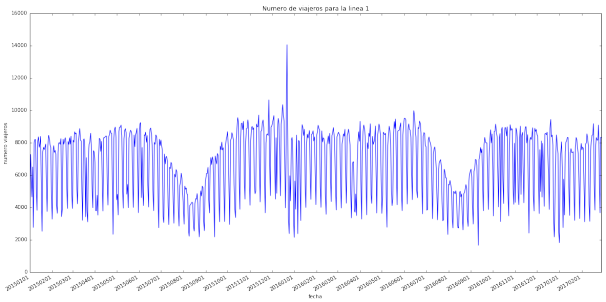


Figura 4-1.: Gráfica Línea 1.

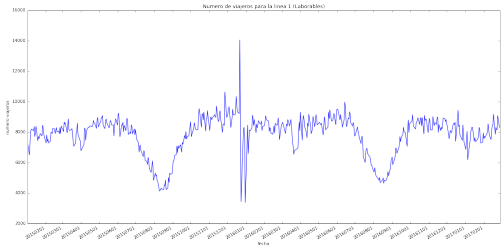


Figura 4-2.: Gráfica Línea 1 Laborables.

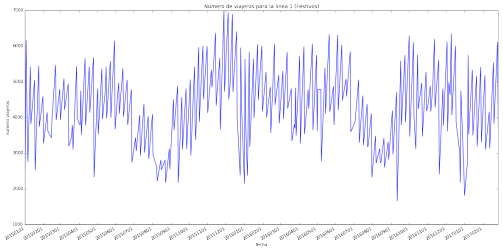


Figura 4-3.: Gráfica Línea 1 Festivos.

- **Tramos horarios:** vimos que al separar nuestras líneas en cuatro tramos horarios (0-6h,6-12h,12-18h y 18-24h) era mejor para nuestras predicciones, ya que hay tramos con un comportamiento diferente para cada una de las líneas dependiendo de su propia naturaleza. En las siguientes imágenes podemos ver dos tramos bien diferenciados de la línea 1:

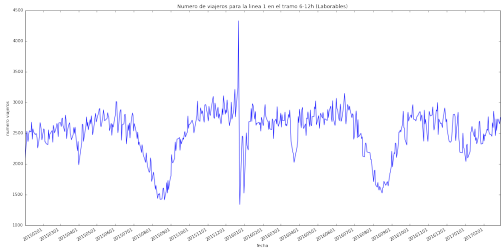


Figura 4-4.: Gráfica Linea 1 tramo 06-12h.

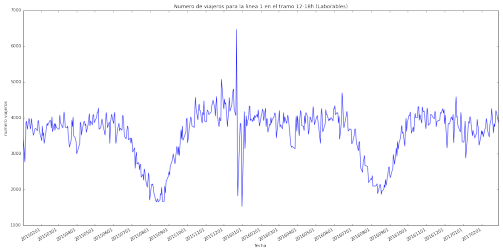


Figura 4-5.: Gráfica Linea 1 tramo 12-18h.

- **Outliers o Errores:** al representar algunas gráficas nos dimos cuenta de la presencia de outliers o errores, ya que encontrábamos algunos viajeros en líneas diurnas en horas nocturnas, y por otra parte encontrábamos viajeros en líneas nocturnas pero en horas diurnas. Este problema lo detallaremos más en el siguiente apartado. En las siguientes gráficas podemos ver los outliers para una línea diurna en el tramo de 0-6h y para una nocturna en el tramo de 11-14h:

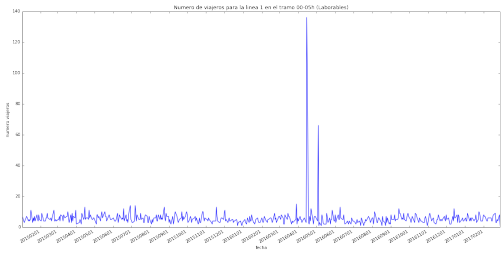


Figura 4-6.: Gráfica Linea 1 Outliers.

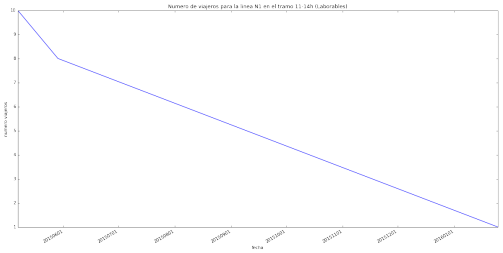


Figura 4-7.: Gráfica Linea N1 Outliers.

4.2. Eliminación de Outliers

Como vimos en el anterior apartado, tenemos algunos outliers en las líneas. Una vez que detectamos esto se lo comunicamos a la EMT y nos dijeron que podían ser errores en la base de datos. Para solventar dicho error generamos una nueva tabla máster (viajeros_tramos_master_nooutliers) sin estos valores, para lo que se realizó un script en Python (7_poda_outliers.hql) que se incluyoó en la carga de los datos. Para ello realizamos las siguientes operaciones para cada conjunto de líneas:

- **Líneas Nocturnas :** para las líneas nocturnas hemos procedido a mantener únicamente los valores que aparezcan entre las 23 y las 07 horas, ya que el resto de valores no corresponden a los horarios de dichas líneas y por lo tanto son outliers.

- **Líneas Trabajo/Otros :** para este tipo de líneas no se ha realizado ninguna operación ya que dichas líneas operan tanto por el día como por la noche.

- **Resto de Lineas :** el resto de líneas son las líneas comunes y que circulan durante el día, por lo que hemos procedido a mantener los valores que sean inferiores a las 02 y superiores a las 06 horas.

4.3. Estudio de las Variables

Seguidamente se ha realizado un estudio más exhaustivo de las líneas y de las variables de las fuentes externas para ver el comportamiento de estas y así poder descubrir algún tipo de patrón.
En los siguientes subapartados vamos a explicar los procesos llevados a cabo.

4.3.1. Estudio Temporal

En primer lugar, hemos visto interesante analizar una línea diurna (en nuestro caso la linea 1) y una nocturna (en nuestro caso la linea N1) para cada mes, día de la semana y tramo horario.

- **Meses :** hemos considerado interesante estudiar el comportamiento de los viajeros para cada mes ya que mediante este estudio podemos ver si el comportamiento se adapta a un comportamiento real. En las siguientes imágenes podemos ver que el comportamiento de estas es muy exacto, ya que los meses que más demanda de viajeros tienen son febrero y diciembre y por el contrario Agosto es el mes que menos viajeros tienen ambas líneas.

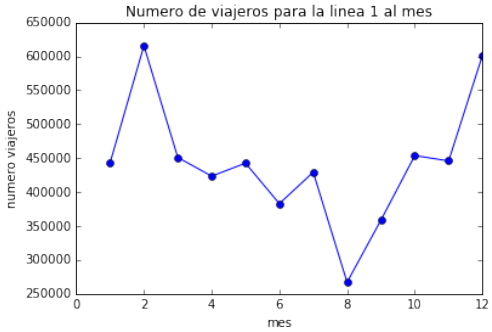


Figura 4-8.: Gráfica Linea 1 viajeros al mes.

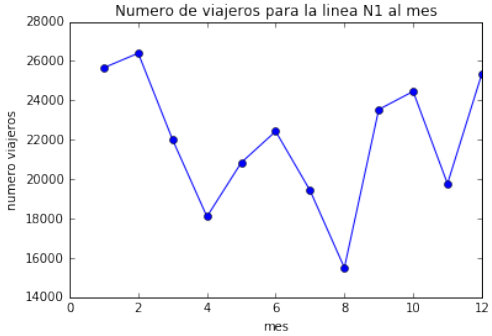


Figura 4-9.: Gráfica Linea N1 viajeros al mes.

- **Días Semana :** otro aspecto a tener en cuenta en nuestro análisis son los días de la semana. En las siguientes imágenes podemos ver que el jueves es el día con menos viajeros y que como era de esperar el lunes es el día más frecuente para la línea diurna y el domingo (Sábado por la noche) para la línea nocturna.

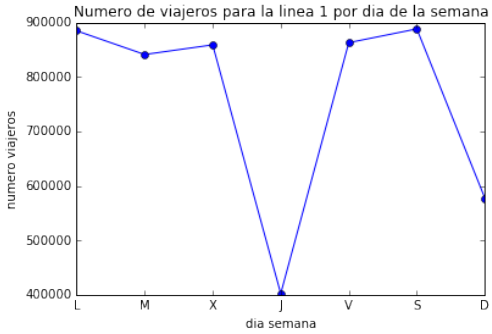


Figura 4-10.: Gráfica Linea 1 viajeros al día.

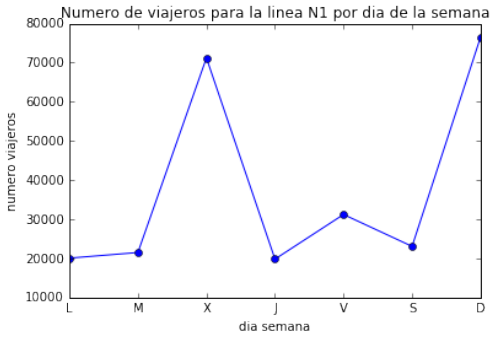


Figura 4-11.: Gráfica Linea N1 viajeros al día.

- **Tramos :** finalmente, otro estudio que hemos realizado es el de ambas líneas por tramos horarios para ver cómo se comportan los viajeros en cada momento. En las siguientes imágenes podemos ver el comportamiento para las dos líneas estudiadas: en lo referente a la línea 1 vemos que las horas de mas tráfico de viajeros están por la mañana y por la tarde, teniendo un pico a las 12 de la mañana y otro a las 18 horas. También podemos ver que el momento en el que decae el número de viajeros es sobre la hora de la comida. Por otro lado la línea N1 tiene un incremento antes de las 02 horas y posteriormente decae teniendo un pequeño pico sobre las 04 horas. Esto es debido a que el primer pico corresponde con las personas que vuelven del trabajo por la noche y el segundo sería un público más joven que saldría por la noche.

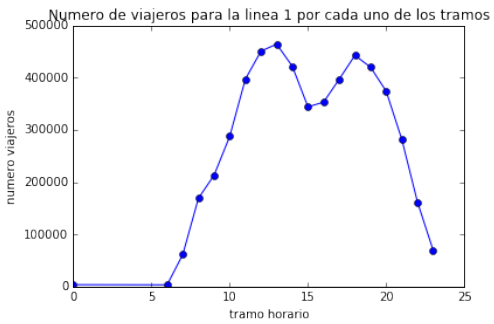


Figura 4-12.: Gráfica Línea 1, viajeros por tramo horario.

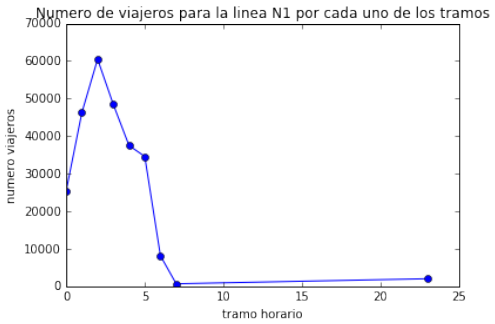


Figura 4-13.: Gráfica Línea N1, viajeros por tramo horario.

El script para realizar este estudio es 2_Auditoria_EstudioTemporal.ipynb.

4.3.2. Estudio de la Demanda

Para comprobar el alcance de nuestro estudio vimos interesante realizar un estudio aproximado de la demanda que tienen las líneas. Para este estudio al igual que para lo visto anteriormente se han seleccionado las líneas 1 y N1.

En primer lugar se ha estudiado a nivel general la demanda para cada una de estas líneas midiendo los siguientes parámetros:

- **Demanda de autobuses :** este estudio nos indica el número medio de autobuses que tiene dicha linea al día. Podemos ver que este es en la línea 1 inferior a 9:

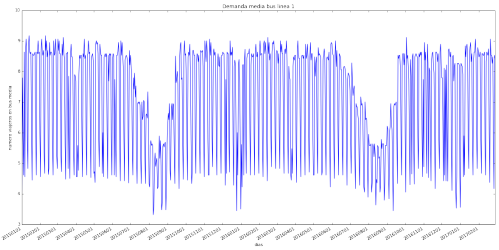


Figura 4-14.: Gráfica de la demanda de autobuses de la Línea 1.

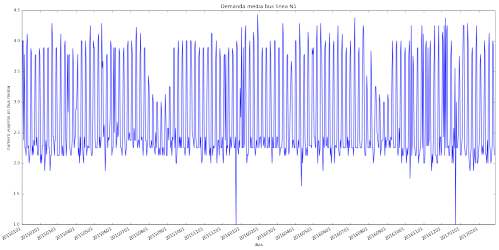


Figura 4-15.: Gráfica de la demanda de autobuses de la Línea N1.

- **Demanda de viajeros :** mediante este estudio podemos ver el número medio de viajeros que transporta cada autobús en cada una de las líneas.

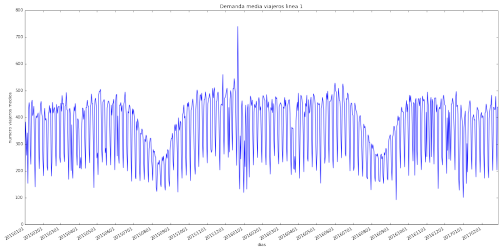


Figura 4-16.: Gráfica de la demanda de viajeros de la Línea 1.

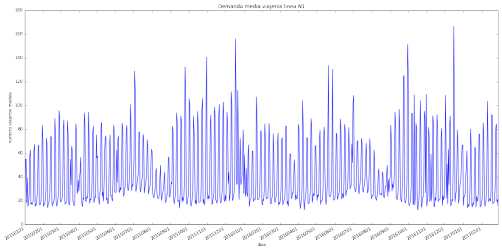


Figura 4-17.: Gráfica de la demanda de viajeros de la Línea N1.

- **Demanda general normalizada :** mediante este estudio podemos ver la demanda media general de viajeros normalizada en cada uno de los autobuses.

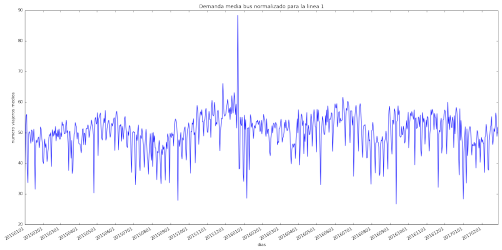


Figura 4-18.: Gráfica de la demanda normalizada de la Línea 1.

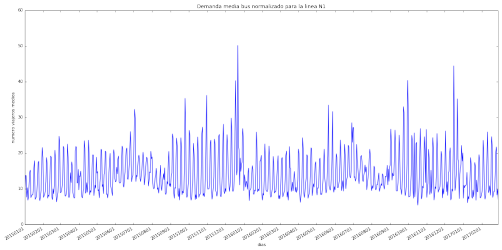


Figura 4-19.: Gráfica de la demanda normalizada de la Línea N1.

Seguidamente hemos estudiado la demanda de la línea 1 en cuatro tramos para ver si la EMT adapta sus autobuses a la demanda no solo por estación sino también por día.

En las siguientes imágenes podemos ver algunos de los tramos para cada una de las demandas estudiadas previamente:

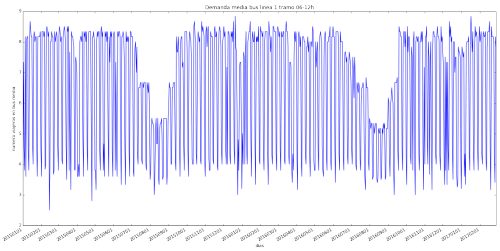


Figura 4-20.: Gráfica de la demanda de autobuses de la Línea 1 tramo 06-12.

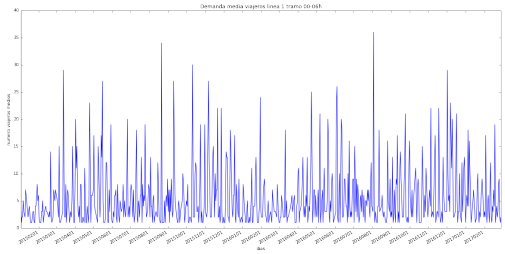


Figura 4-21.: Gráfica de la demanda de viajeros de la Línea 1 tramo 00-06.

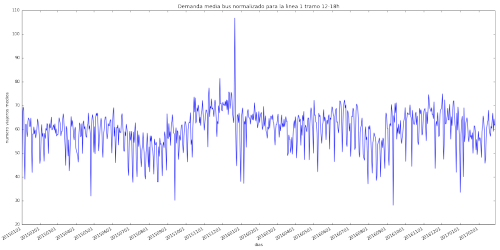


Figura 4-22.: Gráfica de la demanda normalizada de la Línea 1 tramo 12-18.

Como podemos observar parece que la EMT realiza un estudio de su demanda ya que esta es constante para cada tramo de año (invierno, verano, etc). No obstante, su ajuste no óptimo ya que no tienen en cuenta las similitudes entre las líneas, ni tampoco las variables externas (tráfico, Aemet, etc).

En el script 3_Auditoria_EstudioDemanda.ipynb podemos encontrar estos gráficos y su desarrollo.

4.3.3. Estudio de las Variables externas

Otro estudio que hemos realizado ha sido el estudio de nuestras variables procedentes de fuentes externas junto con la demanda de viajeros estudiada anteriormente. Para ello se han estudiado dos líneas que contengan todas las variables procedentes de las fuentes externas (lluvia, tráfico y eventos deportivos), por lo que se ha seleccionado la línea 14 y la línea N22. Hay que indicar que para las líneas nocturnas (en nuestro caso la linea N22) no tenemos eventos deportivos ya que ninguno de estos se desarrolla después de las 23:00h. En las siguientes imágenes mostramos los resultados obtenidos para la línea diurna estudiada (línea 14):

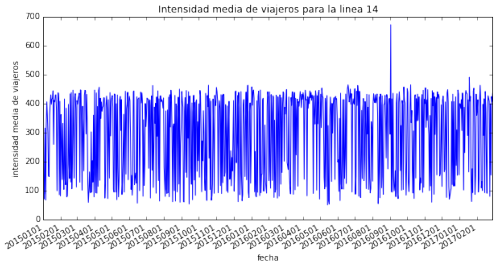


Figura 4-23.: Gráfica de la demanda de viajeros línea 14.

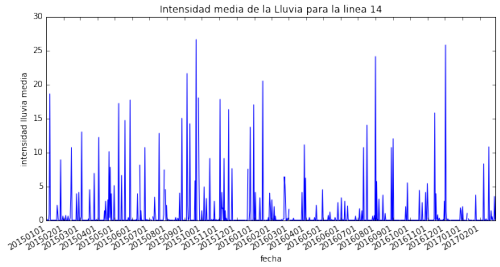


Figura 4-24.: Gráfica de la intensidad de lluvia línea 14.

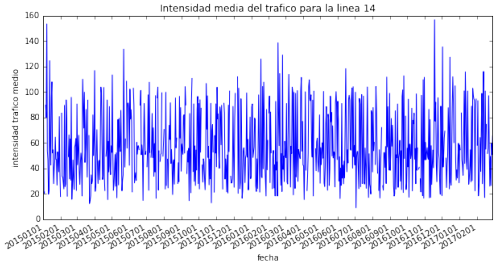


Figura 4-25.: Gráfica de la intensidad del tráfico línea 14.

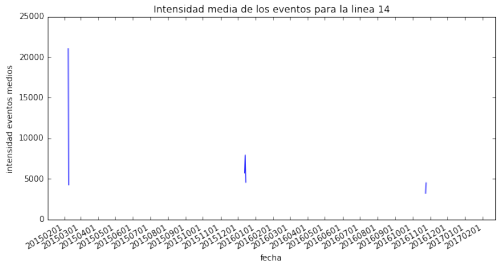


Figura 4-26.: Gráfica de la intensidad de los eventos línea 14.

En las siguientes imágenes podemos observar las mismas gráficas pero en este caso para la línea nocturna estudiada (linea N22):

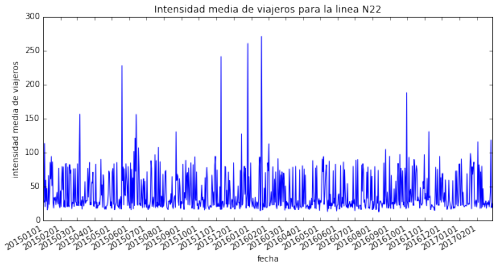


Figura 4-27.: Gráfica de la demanda de viajeros línea N22.

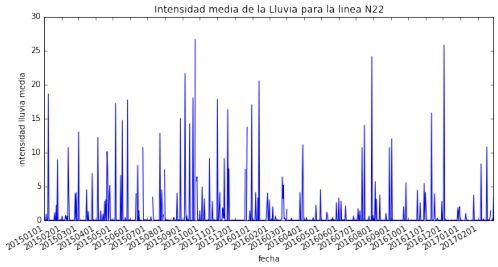


Figura 4-28.: Gráfica de la intensidad de lluvia línea N22.

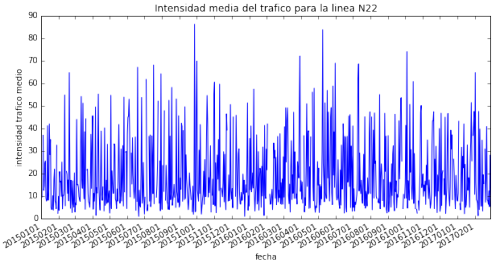


Figura 4-29.: Gráfica de la intensidad del tráfico línea N22.

Se observa que las variables externas que más afectan a la demanda de viajeros son la intensidad de la lluvia y del tráfico, de tal forma que si estas aumentan hacen que la demanda de viajeros crezca y por lo tanto son variables que tenemos que tener en cuenta a la hora de desarrollar nuestro modelo analítico.

En el script 4_Auditoria_EstudioVariables.ipynb podemos encontrar el desarrollo de lo explicado en este apartado.

4.4. Detección de Líneas Raras

Una vez que tenemos todos los datos analizados vimos la necesidad de tener en cuenta ciertas líneas raras"que tienen un comportamiento extraño respecto al resto de líneas. Detectamos que ciertas líneas solo eran usadas unos pocos días al año, como por ejemplo la línea 180 que se trata de la línea que va hacia la Caja Mágica.

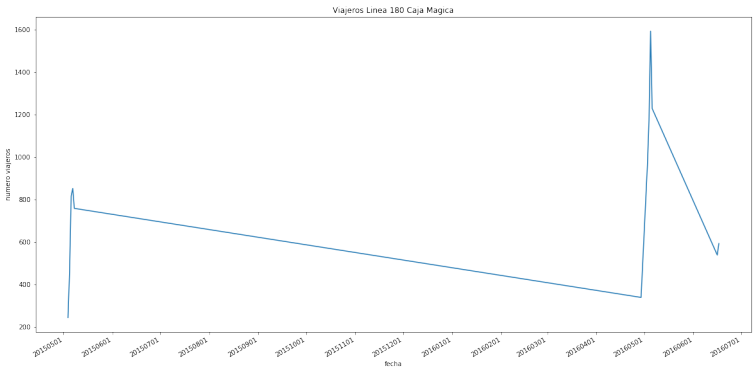


Figura 4-30.: Gráfica Linea 180 (Laborables).

Otras líneas únicamente tenían datos en una etapa corta dentro de nuestro histórico y consideramos que por ello no deberían incluirse en nuestro estudio.

Por lo tanto no hemos eliminado dichas líneas de nuestra base de datos, ya que si estas siguen siendo utilizadas podrían servirnos en un análisis futuro. Tan solo no han sido consideradas por ejemplo para la fase de clustering que veremos más adelante ya que consideramos que no aportan suficiente información en este momento.

5. Desarrollo de los modelos predictivos

En este capítulo vamos a ver los modelos analíticos que se han utilizado para los diversos estudios analíticos realizados.

5.1. Segmentación de líneas de autobuses

Hemos optado por realizar un clustering de las líneas de autobuses para ver qué líneas son más similares y así poder aplicar los mismos tipos de mejoras en estos segmentos.

En primer lugar, y como vimos anteriormente en el apartado de fuentes de datos, todas las líneas que tenemos se dividen en siete segmentos (Común, Nocturno, Trabajo, Servicio Especial, Circular, Microbús y Universitaria) que ya hemos explicado anteriormente. A pesar de ya tener categorizadas dichas líneas, se ha realizado un clustering de todas para verificar si hay líneas de otros sectores que se relacionan entre sí y así poder estudiarlas más en profundidad. Para dicho clustering se ha considerado la separación entre laborables y festivos (fines de semana y festivos), ya que como vimos en el apartado anterior, el comportamiento de cada línea para los laborables y para los festivos es muy diferente y merece un estudio especial. Para la realización de nuestro clustering, en primer lugar tenemos que mencionar que se han utilizado tres técnicas: SAX-K-means, K-means y Clustering Jerárquico.

5.1.1. SAX-K-Means

En primer lugar, hemos realizado un clustering de dichas líneas empleando la técnica SAX (Symbolic Aggregate approXimation), mediante la cual discretizamos las series temporales en letras, en donde cada letra nos indica un segmento de la serie temporal de tal forma que si dicho segmento es muy similar en distintas series, estas tendrán asignada la misma letra [**6**].

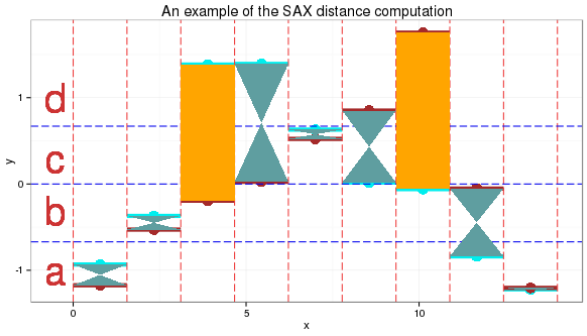


Figura 5-1.: Representación SAX.

Una vez que tenemos discretizadas las series temporales, realizamos un algoritmo de clustering K-means, mediante el cual podemos segmentar las líneas en diversos segmentos. En nuestro caso hemos elegido k=8 tras realizar varias pruebas con diferentes tamaños de k. En las siguientes imágenes podemos ver los 8 clusters generados para los días laborables:

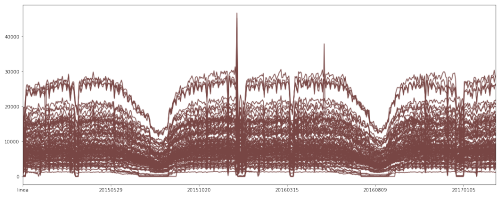


Figura 5-2.: Cluster SAX 1.

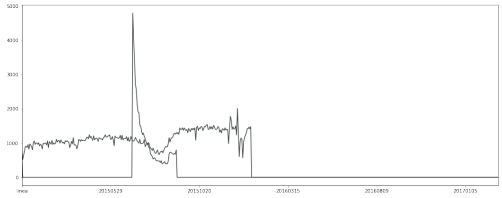


Figura 5-3.: Cluster SAX 2.

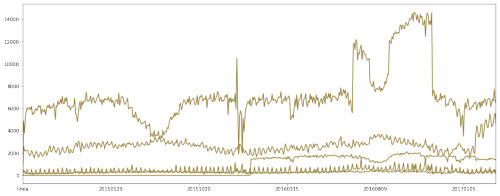


Figura 5-4.: Cluster SAX 3.

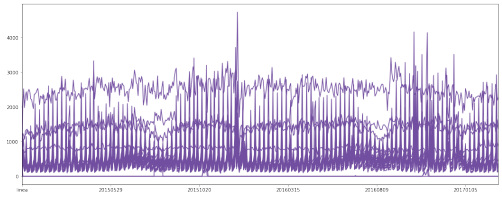


Figura 5-5.: Cluster SAX 4.

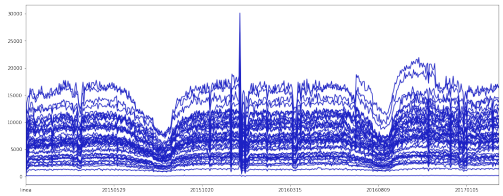


Figura 5-6.: Cluster SAX 5.

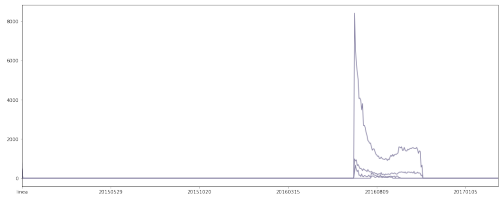


Figura 5-7.: Cluster SAX 6.

Figura 5-8.: Cluster SAX 7.

Figura 5-9.: Cluster SAX 8.

Podemos ver que los clusters no realizan una segmentación adecuada de las líneas ya que hay clusters donde hay líneas con comportamientos muy diferentes. Una vez que vimos dichas diferencias procedimos a normalizar las series temporales para ver si mejoraban los clusters y nos dimos cuenta que sí que nos mejoraban los resultados pero aún así no era suficiente ya que teníamos líneas muy diferentes en un mismo cluster y no éramos capaces de justificar dicha agrupación.

5.1.2. K-Means y Clustering Jerárquico

Dados los resultados vistos anteriormente, hemos visto interesante aplicar el método K-means y Jerárquico sin aplicar previamente el algoritmo SAX. Hay que indicar que cuando se tienen una gran cantidad de períodos en las series temporales la mejor opción es realizar SAX o emplear un algoritmo basado en redes neuronales. En nuestro caso, una vez probados los dos algoritmos anteriores, hemos probado a aplicar a cada una de nuestras líneas para cada uno de los días el algoritmo K-means en primer lugar y posteriormente un algoritmo de clustering jerárquico[**7**], de tal forma que nuestras variables son todos los días en nuestra base de datos cuyo valor son el número de pasajeros para cada día agregados. En los siguientes apartados vamos a explicar cada una de las fases realizadas:

- **Eliminación de Líneas Raras":** en primer lugar, hemos procedido a eliminar de nuestro estudio las líneas raras"que hemos estudiado anteriormente (apartado 3.4), ya que estas líneas nos pueden empeorar nuestro estudio y por lo tanto no nos aportan valor y merecen ser estudiadas por separado.
- **Normalización :** seguidamente, hemos realizado una normalización de los dos dataset (laborables y festivos) empleando el algoritmo StandarScaler, es decir, restamos cada

valor por la media y lo dividimos por la desviación estándar. Esto es para que todas las líneas estén representadas en la misma escala y por lo tanto no tomen más importancia las líneas que tienen una escala de demanda mayor que el resto.

- **Clustering K-Means :** una vez tenemos preprocesados los datos de cada una de las líneas, se ha realizado un clustering empleando el algoritmo K-means, en donde las variables de nuestro modelo serían los valores de cada uno de los días que disponemos para nuestro estudio y cada una de las líneas de la base de datos corresponden con la línea que deseamos segmentar. En cuanto al número de clusters hemos observado que el K que mejor segmenta las líneas es 6. Esto lo hemos realizado mediante la ejecución con varias K y viendo dónde nuestro algoritmo empezaba a separar demasiado las líneas. En dicho momento nos quedábamos con la K que mejor nos separaba las líneas de una forma coherente siguiendo nuestro conocimiento del negocio. Por lo tanto, tendremos 6 clusters diferentes:

Cluster	Número Líneas	Líneas
Cluster 0	135 líneas	1,2,4,5,6,7,8,9,11,12,14,15,16,17,18,19,20,21,22,23,25,27, 28,29,30,31,34,35,36,38,39,40,41,42,43,44,45,46,47,48,49, 50,51,52,53,55,56,58,59,60,61,62,63,64,65,67,68,69,70,71, 72,73,74,75,76,77,78,82,83,85,86,87,90,91,92,93,96,99,100, 104,105,106,107,112,113,114,115,116,117,120,121,122,123, 125,126,127,128,129,131,132,133,134,135,137,138,139,140, 144,146,147,150,151,152,153,155,156,160,161,162,172,173, 174,176,178,210,215,401,402,403,451,455,456,481,601,602
Cluster 1	9 líneas	101,200,203,404,718,731,756,766,767
Cluster 2	26 líneas	501,502,503,504,505,506,507,508,509,510,511,512,513,514, 515,516,517,518,519,520,521,522,523,524,525,526
Cluster 3	9 líneas	33,110,247,702,704,728,729,730,799
Cluster 4	7 líneas	37,171,527,714,715,716,788
Cluster 5	32 líneas	3,10,24,26,32,54,57,66,79,81,102,103,108,109,111,118,119, 124,130,136,141,142,143,145,148,149,177,310,452,453,454, 457

Tabla 5-1.: Clustering K-means Laborables

Cluster	Número Líneas	Líneas
Cluster 0	27 líneas	3,10,24,26,33,37,41,47,56,57,59,76,79,86,107,118,119, 130,135,142,143,145,148,171,177,372,481
Cluster 1	118 líneas	1,2,4,5,6,7,8,9,11,12,14,15,16,17,18,19,20,21,23,25,27, 28,29,30,31,32,34,35,36,38,39,40,42,43,44,45,46,48,49,50, 51,52,53,54,55,58,60,61,62,63,64,65,66,67,68,69,70,71,72, 73,74,75,77,78,81,82,85,87,100,102,103,104,105,106,109, 111,112,113,114,115,116,120,121,122,123,124,125,126,127, 128,131,132,133,134,136,137,138,139,140,141,144,146,147, 150,152,153,155,160,161,162,173,174,176,178,210,215,402, 403
Cluster 2	10 líneas	101,180,200,203,404,527,729,731,756,767
Cluster 3	41 líneas	22,83,93,108,117,129,149,151,156,247,310,401,453,501, 502,503,504,505,506,507,508,509,510,511,512,513,514,515, 516,517,518,519,520,521,522,523,524,525,526,601,602
Cluster 4	4 líneas	714,715,716,788
Cluster 5	8 líneas	110,172,702,704,728,732,742,799

Tabla 5-2.: Clustering K-means Festivos

Como podemos ver, las líneas se agrupan bien tanto para los laborables como para festivos, ya que tenemos las líneas diurnas en los cluster 0 (laborables) y cluster 1 (festivos). También podemos observar que las líneas nocturnas se agrupan en los cluster 4 (laborables) y 3 (festivos). También podemos ver en los clusters restantes las líneas raras"que tienen un gran número de paradas como la línea 22, o un cluster de líneas que operan todo el año de una forma constante como son las del aeropuerto (cluster 1 en laborables y cluster 2 en festivos). Esta explicación la veremos mejor cuando expliquemos los resultados con el clustering jerárquico.

Adicionalmente, para comprobar que los clusters se realizaban correctamente, hemos representado cada una de las líneas normalizadas por cada cluster junto con sus respectivos centroides. A continuación mostramos cada uno de los clusters para laborables:

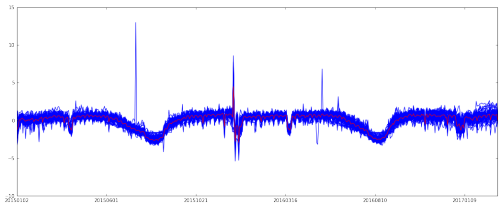


Figura 5-10.: Cluster 0 KMeans Laborables.

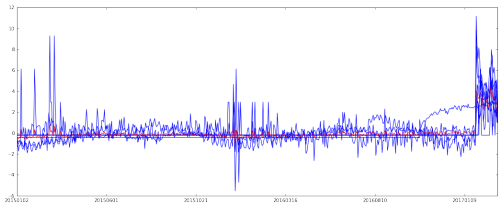


Figura 5-11.: Cluster 1 KMeans Laborables.

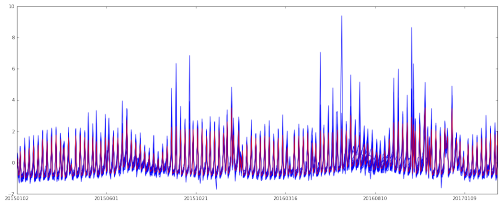


Figura 5-12.: Cluster 2 KMeans Laborables.

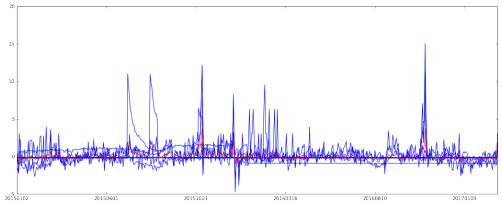


Figura 5-13.: Cluster 3 KMeans Laborables.

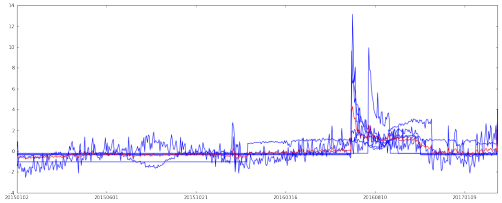


Figura 5-14.: Cluster 4 KMeans Laborables.

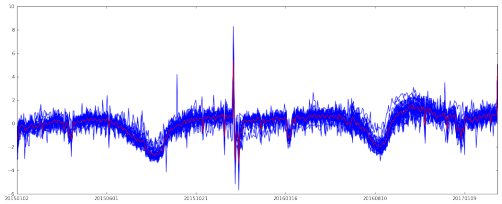


Figura 5-15.: Cluster 5 KMeans Laborables.

Podemos ver que el centroide (línea roja) representa bien a todas las líneas de dicho cluster y que además todas las líneas de cada cluster son muy similares, por lo que podemos concluir que el clustering realizado segmenta correctamente y de la forma esperada todas las líneas.

- **Matriz de Correlación :** adicionalmente, para comprobar las correlaciones entre las líneas de los clusters mostrados anteriormente hemos calculado la matriz de correlación. En primer lugar se han ordenado las líneas en orden en el que aparecen en los clusters y seguidamente se ha calculado la correlación entre ellas mediante los datos diarios. Para ver mejor dicha correlación hemos pintado esta matriz en un mapa de calor. A continuación mostramos este mapa de calor para el clustering de laborables:

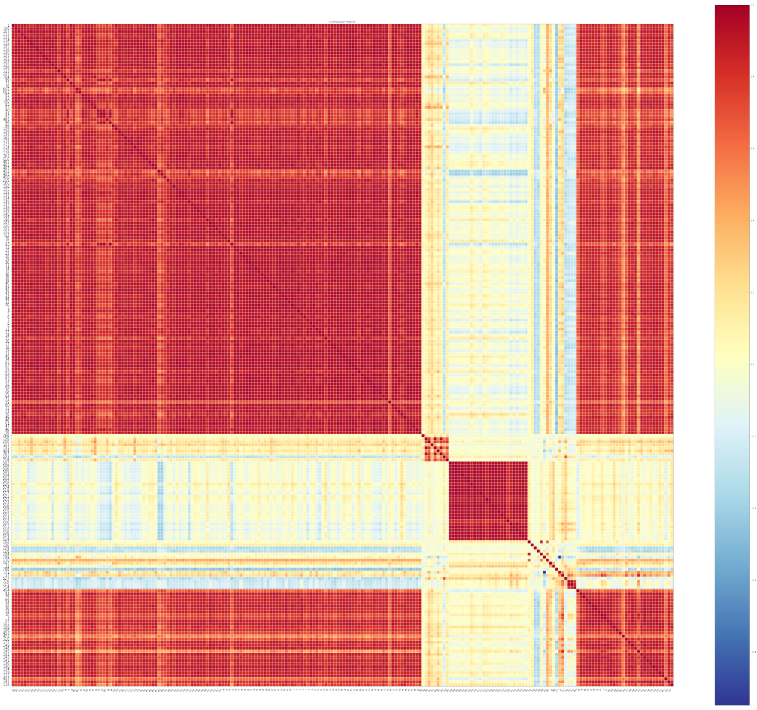


Figura 5-16.: Matriz de correlación Laborables.

Podemos observar que las líneas más correlacionadas están representadas en rojo y las menos correlacionadas en azul. Las líneas que forman cuadrados corresponden a los clusters anteriormente definidos, en donde se puede ver que coinciden con los clusters vistos obtenidos mediante K-means.

- **Clustering Jerárquico :** para comprobar de una forma más visual el clustering realizado mediante K-means se ha realizado un clustering jerárquico. El algoritmo elegido ha sido el dado por la librería sciPy *hierarchical clustering* con distancia euclídea. Mediante dicho método hemos visto de una forma más visual cómo se distribuyen las líneas en función de sus viajeros. A continuación, vamos a ir viendo por ramas cada uno de los clusters:

  - **Rama 1 :** en esta rama podemos ver que tenemos cuatro conjuntos de líneas bien diferenciados: en el primer segmento tenemos las líneas que van hacia el aeropuerto. Como vemos el modelo ha identificado correctamente estas líneas ya que tienen una alta frecuencia de viajeros durante todos los días incluso en verano. En el segundo subconjunto tenemos las líneas temporales que vemos que han sido correctamente detectadas por el clustering. Esto es debido a que estas tienen un número bajo de viajeros. Finalmente tenemos el subconjunto de las líneas nocturnas donde su afluencia se marca en tramos nocturnos.

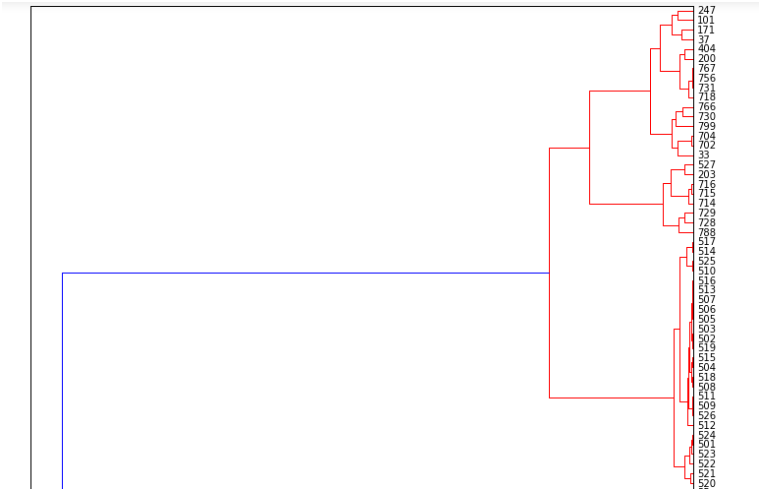


Figura 5-17.: Dendrograma Laborables Rama 1.

- **Rama 2 :** esta rama hace referencia a líneas con un largo recorrido y que normalmente pasan por el centro de Madrid. Estas líneas, al tener un recorrido mayor al resto, suelen estar contenidas en un único cluster.

Figura 5-18.: Dendrograma Laborables Rama 2.

- **Rama 3:** en esta rama nos encontramos líneas que pasan por el centro de Madrid que tienen paradas en común y por lo tanto el número de viajeros es muy similar. Podemos ver que también tenemos líneas como la 69 que se trata de una línea circular y que también pasa por el centro de Madrid.

Figura 5-19.: Dendrograma Laborables Rama 3.

- **Rama 4 :** en esta rama tenemos líneas que generalmente desembocan o nacen en la periferia de Madrid, como puede ser líneas de la zona de Vallecas, las de la zona de Pitis, etc. Estas líneas son frecuentadas principalmente por viajeros que van hacia el centro o hacia ubicaciones como Atocha.

Figura 5-20.: Dendrograma Laborables Rama 4.

- **Rama 5 :** en esta rama tenemos las líneas raras (cementerio de la Almudena, etc) y algunas líneas universitarias. Estas son líneas que no siguen ningún patrón en común con otras líneas y que por tanto se agrupan en una única rama.

Figura 5-21.: Dendrograma Laborables Rama 5.

Podemos ver que en cada una de las ramas tenemos otras subramas. A medida que vamos bajando por el árbol veremos líneas que tienen más paradas en común, viajeros, etc. En el Anexo II podemos encontrar el dendrograma completo, que está dividido en tres partes para una mejor visualización.

Los scripts para esta parte del proyecto los podemos encontrar en la carpeta 4_Analitica\1_clustering.

5.2. Predicción de la Demanda

5.2.1. Descripción

Por último, se ha decido implementar un estudio para la predicción de la demanda. Hay que indicar que este es el objetivo principal de nuestro proyecto, ya que se detectó una gran necesidad por parte de EMT para controlar y mejorar cada una de las líneas mediante sus históricos y mediante fuentes externas. Para la realización de la predicción de la demanda nos hemos centrado en primer lugar en el uso de las técnicas de ARIMA [**3**] (Modelo Autorregresivo Integrado de Media Móvil), el cual es un modelo estadístico que utiliza variaciones y regresiones de datos estadísticos con el fin de encontrar patrones para una predicción hacia el futuro. En segundo lugar, hemos hecho uso de técnicas de Deep Learning para mejorar nuestra predicción de la demanda por medio del uso de redes neuronales profundas empleando el algoritmo LSTM [**13**].

5.2.2. Implementación

En cuanto a su implementación, hay que indicar que se ha optado por realizar las siguientes fases para cada una de sus dos implementaciones:

- **Modelo ARIMA:**
  - **Fase de Carga:** En primer lugar se han cargado los datos desde Hive empleando Pyspark en donde se han filtrado línea a línea por tramo horario y se ha preprocesado previamente el dataset. Los datos se han cargado desde la tabla viajeros_tramos_master_nooutliers que, como hemos indicado anteriormente, se encuentra libre de outliers en cada una de las líneas a estudiar. Adicionalmente se han creado dos variables derivadas que son el número de la semana (es decir, si estamos en la primera, segunda, tercera o cuarta semana del mes) y el día de la semana (siendo 1 el Lunes, 2 el Martes, etc.)
  - **Fase de Predicción:** Una vez que tenemos filtrado y preprocesado el dataset a estudiar de cada una de las líneas, procedemos a la fase de predicción. Para dicha fase hemos usado en primer lugar el modelo de auto-ARIMA que dispone R debido a que nuestros datos los cargamos desde un notebook de Pyspark. Hemos procedido a insertar código R en un notebook de Python mediante la librería Rpy2 mediante la cual podemos insertar celdas de R en un notebook de Python y trabajar con ambos lenguajes en un mismo notebook. En primer lugar, tenemos que pasar nuestro dataset en formato Pandas al formato de dataset empleado por R. Una vez tenemos dicho dataset aplicamos las fases de predicción de auto-ARIMA: primero calculamos el ACF (AutoCorrelation Function) y el PACF (Partial AutoCorrelation Function) que nos indican los coeficientes de autocorrelación de nuestra serie temporal mediante la cual podemos ver las correlaciones lineales entre los días de la serie.

    Seguidamente añadimos una serie temporal con las variables externas. Estas variables las meteremos como el parámetro xreg en nuestro modelo ARIMA y condicionarán la predicción de la serie temporal.

    Finalmente realizamos la separación entre training y test. Para training hemos tomado todos los días de 2015 y 2016, mientras que para test hemos tomado los días de 2017. En la siguiente imagen podemos ver la predicción de nuestro modelo:

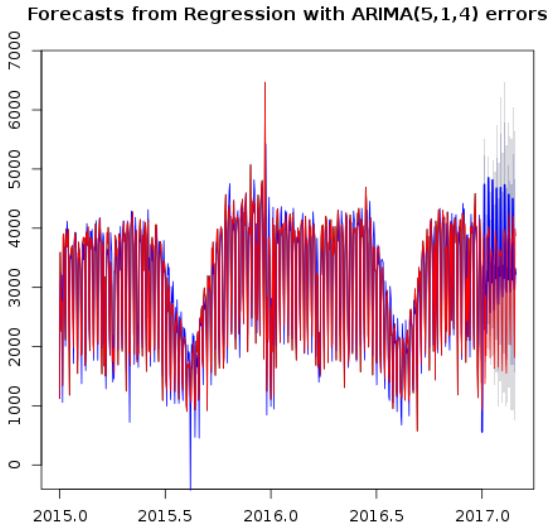


Figura 5-22.: Predicción línea 1 con ARIMA.

- **Fase de Evaluación:** mediante este modelo hemos obtenido un scoring de 75 % de acierto. No obstante cuando procedemos a realizar zoom en la serie predicha, podemos ver que esta no está prediciendo correctamente ya que posee un gran error en la predicción. En la siguiente imagen podemos ver el error en la predicción que acabamos de mencionar:

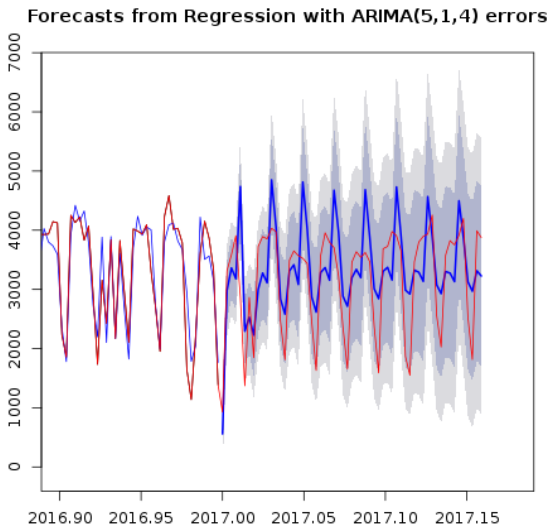


Figura 5-23.: Predicción línea 1 con ARIMA (zoom).

- **Modelo Deep Learning (LSTM):**

  - **Fase de Carga:** para la fase de carga de datos se ha generado un notebook para extraer los datos en local, es decir generar varios ficheros csv para cada una de las líneas a estudiar. Esto es debido a que la instalación de Keras y sus dependencias en el cluster era muy compleja y se salía de los objetivos de nuestro proyecto. El único inconveniente que tenemos es la realización de nuestro estudio en local. No obstante, con los datos que manejamos por línea es abarcable.

  - **Fase de Predicción:** en primer lugar una vez que tenemos los cuatro ficheros (uno por cada tramo horario) para cada una de las líneas a estudiar, tenemos que normalizar los datos. Sólo se ha procedido a normalizar la demanda de viajeros, ya que es la única que tiene valores extremos y es la usada para la predicción. En cuanto a las variables externas, al tomar valores continuos y que no son usados como variable objetivo no es necesario normalizarlas. Para la normalización de los datos de la demanda se ha procedido a utilizar las siguientes fórmulas:

    Transformación:

    $$a_{n+1} = \ln \frac{o_{n+1} + \epsilon}{o_n + \epsilon}$$

    Transformación inversa:

    $$\hat{o}_{n+1} = o_n \cdot \exp \hat{a}_{n+1}$$

    Donde ϵ es una constante con un valor muy cercano a cero para evitar los ceros tanto en el numerador como en el denominador de la división dentro del logaritmo ($\epsilon = 1e - 11$). En la primera fórmula podemos ver que en esta normalización estamos empleando logaritmos teniendo en cuenta siempre el instante de tiempo previo y posterior. La segunda fórmula la utilizamos para realizar la transformación inversa a la normalización, para que tras la predicción realizada por nuestro sistema podamos volver a los valores originales.

    Una vez que hemos realizado la normalización de los datos procedemos a construir el sistema predictivo. Para ello hemos empleado LSTM mediante el uso de la librería Keras [**27**]. En la red neuronal hemos utilizado una configuración con 5 neuronas ya que tras varias pruebas con diferentes números es la que mejor se adaptaba a nuestras necesidades, y un lookback de 4 para evitar que la red tome únicamente el valor anterior al que se va a predecir. En la siguiente imagen podemos ver la predicción de la serie temporal:

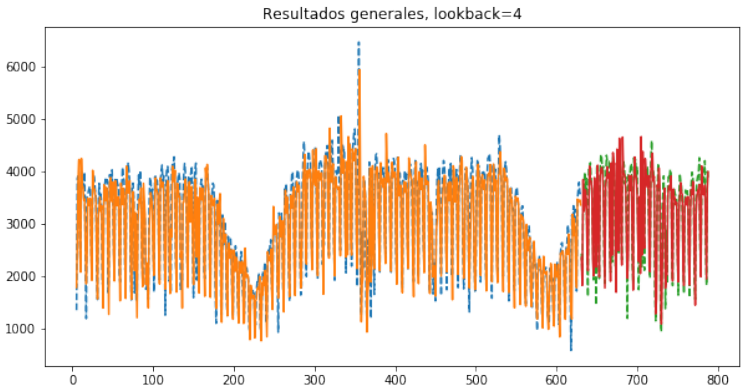


Figura 5-24.: Predicción mediante LSTM.

- **Fase de Evaluación:** una vez que hemos realizado la predicción tenemos que evaluar si nuestro resultado es el adecuado. Para ello procedemos a aplicar la transformación inversa a la predicción realizada por el algoritmo y representamos las predicciones en una gráfica ampliada para ver más de cerca los resultados de dicha predicción:

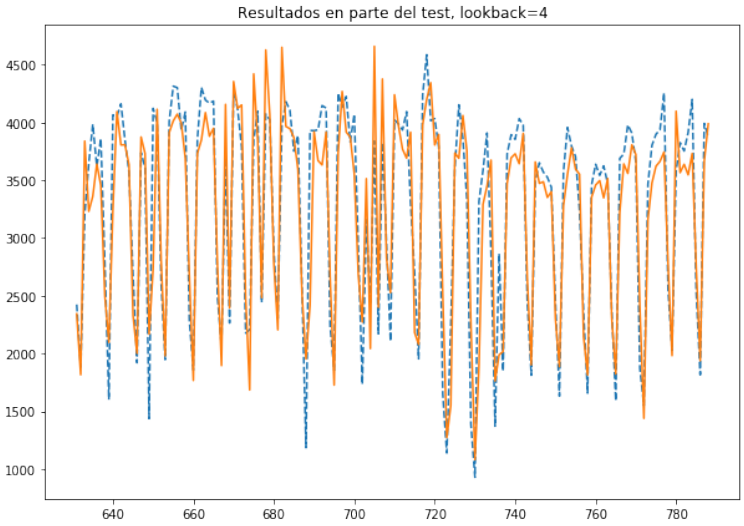


Figura 5-25.: Predicción mediante LSTM (ampliada).

El valor real está pintado en azul, el valor predicho en training en naranja y el valor predicho en test en rojo. Podemos ver que la predicción se ajusta bastante a la realidad salvo en algunos casos que es muy difícil llegar a predecir dichos valores ya que son picos extraños.

Finalmente, para cuantificar la bondad de nuestro modelo hemos calculado el RMSE (error cuadrático medio) para calcular el error de nuestro modelo. Hemos

representado los RMSE obtenidos en la fase de training y validación de nuestra red neuronal para ver dónde converge y parar el algoritmo antes de que se llegue a producir un OverFitting:

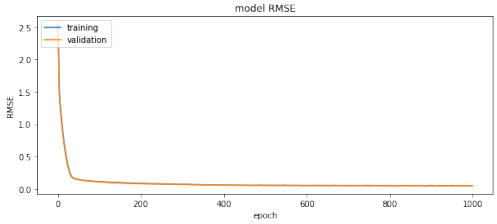


Figura 5-26.: RMSE para la predicción con Keras.

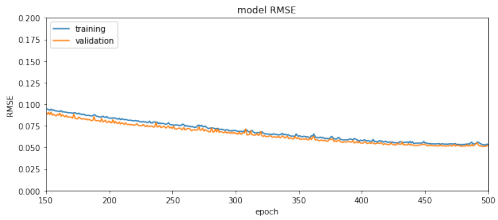


Figura 5-27.: RMSE para la predicción con Keras (Zoom).

Podemos ver que cerca de 100 épocas el RMSE se mantiene y por lo tanto seguir entrenando el modelo podría provocar OverFitting, por lo que un buen valor de epoch sería entre 50 y 100.

Finalmente y como podemos observar, el modelo que mejor predice la demanda es el basado en LSTM ya que podemos ver que tanto su precisión como su evaluación por medio de las gráficas representadas se ajusta más a la realidad. En la siguiente tabla podemos ver la comparativa entre los resultados obtenidos por ambos modelos:

Modelo	RMSE Medio en Training	RMSE Medio en Test
ARIMA (Sin Externas)	299.41	1216.20
ARIMA (Con Externas)	350.98	713.97
LSTM (Sin Externas)	411.82	515.25
LSTM (Con Externas)	271.13	280.20

Tabla 5-3.: Comparativa ARIMA vs LSTM

Podemos ver que efectivamente, el modelo de LSTM con variables externas es mejor que el resto, ya que por una parte su RMSE es menor, y por otra tiene menos sobreajuste (su error en training es muy similar al error en test).

6. Diseño del dashboard de usuario

Adicionalmente se ha desarrollado un dashboard para que el usuario final pueda realizar consultas simples a nuestro sistema y a todos sus datos de una forma más amigable y visual. El sistema elegido ha sido Tableau, ya que es un sistema que nos aporta una gran flexibilidad a la hora de crear dashboards sin tener que desarrollar una gran cantidad de código. A continuación vamos a explicar cada una de las partes en las cuales se divide este dashboard:

- **Visualización del histórico de datos:** en esta pantalla podemos ver una visualización previa del histórico de los datos. Destacan los siguientes gráficos:
  - **Gráfico GTFS:** este gráfico nos muestra el mapa de las líneas que tenemos cargadas en nuestra base de datos, para ello nos hemos descargado los datos GTFS (Google Transit Feed Specification) que se trata de un zip con una serie de ficheros mediante los cuales podemos representar las formas de las líneas y sus paradas.

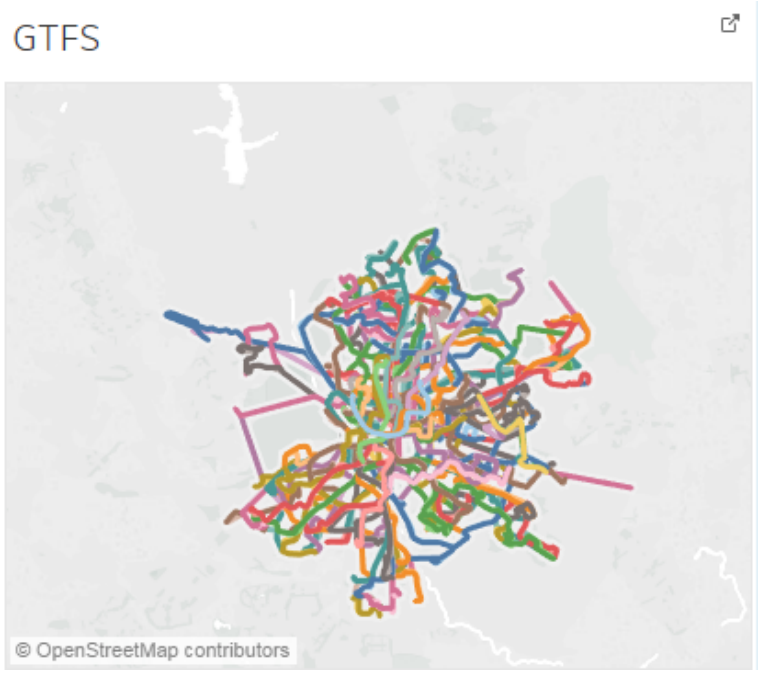


Figura 6-1.: Dashboard: Gráfico GTFS.

- **Gráfico Tipos de Billete:** se ha creado un gráfico que nos permite ver de una forma más gráfica los distintos tipos de billete en nuestro histórico. Este gráfico es muy útil si por ejemplo queremos saber qué tipos de billetes se mueven por una cierta línea y en un determinado momento, de tal forma que podemos ver si en dicha línea viajan un mayor número de parados, jóvenes, mayores, etc.

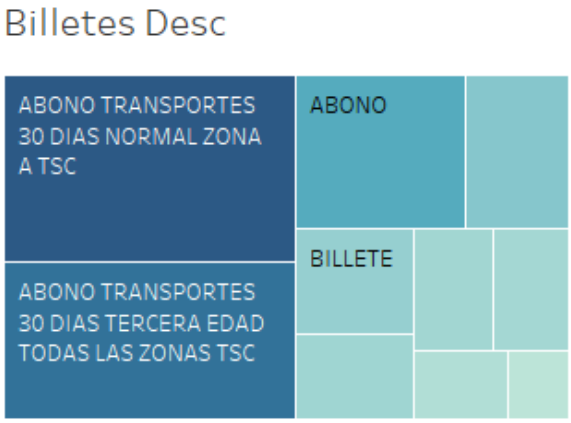


Figura 6-2.: Dashboard: Tipos de Billete por Línea.

- **Histogramas:** finalmente hemos implementado un histograma para ver de una forma más visual el número de viajeros por cada tipo de billete.

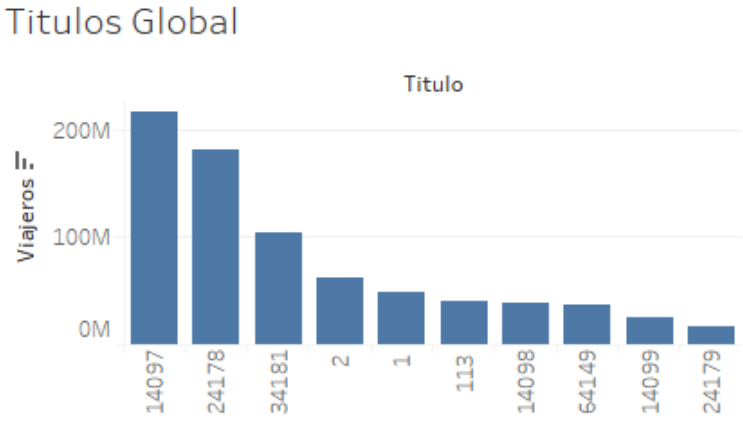


Figura 6-3.: Dashboard: Histograma del número de viajeros por billete.

- **Visualización del clustering:** se han implementado unos gráficos para visualizar el clustering de las líneas realizado. Se ha optado por un gráfico que representa correctamente todos los clusters de las líneas estudiadas mediante el tamaño de las líneas que posee cada cluster para el cluster de laborables y para el de festivos.

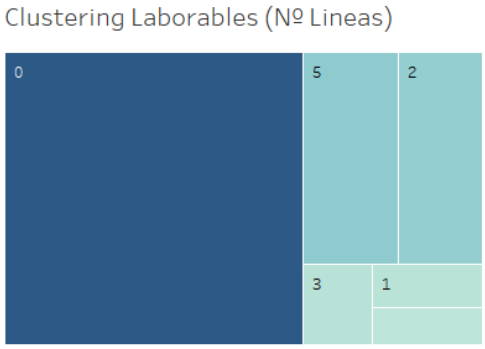


Figura 6-4.: Dashboard: Cluster Laborables.

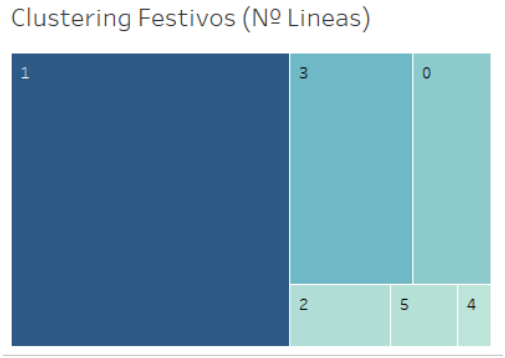


Figura 6-5.: Dashboard: Cluster Festivos.

Adicionalmente, se han realizado dos tablas para ver las líneas y la media de viajeros de cada una de ellas en cada cluster visto en el anterior gráfico.

Tabla_Clust_Lab

Linea (Clust..	Tipo Linea	Avg Viajeros La..	
1	Comun	214,835464880	Abc
2	Comun	320,318662320	Abc
3	Comun	276,864413786	Abc
4	Comun	217,987018297	Abc
5	Comun	187,954659279	Abc
6	Comun	404,534293623	Abc
7	Comun	135,043423995	Abc
8	Comun	230,431547311	Abc
9	Comun	357,418132264	Abc
10	Comun	422,838222406	Abc
11	Comun	152,170875290	Abc

Figura 6-6.: Dashboard: Tabla Cluster Laborables.

Tabla_Clust_Fest

Linea (Clust..	Tipo Linea	Avg Viajeros Fe..	
1	Comun	120,667001339	Abc
2	Comun	165,047849462	Abc
3	Comun	146,514754098	Abc
4	Comun	89,163337772	Abc
5	Comun	78,835428305	Abc
6	Comun	257,461892986	Abc
7	Comun	41,659812868	Abc
8	Comun	92,904745724	Abc
9	Comun	166,921286405	Abc
10	Comun	204,907763173	Abc
11	Comun	45,727159656	Abc

Figura 6-7.: Dashboard: Tabla Cluster Festivos.

- **Predicción de las líneas:** se ha desarrollado una serie de gráficas para representar las dos predicciones de viajeros realizadas en nuestro estudio. Hay que indicar que estas han sido separadas por tramo horario tal y como aparecería en un caso de uso real. En las siguientes imágenes podemos observar las dos gráficas, una para el training y otra para el test realizado:

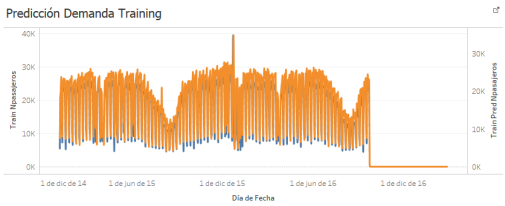


Figura 6-8.: Dashboard: Gráfica Predicción Training.

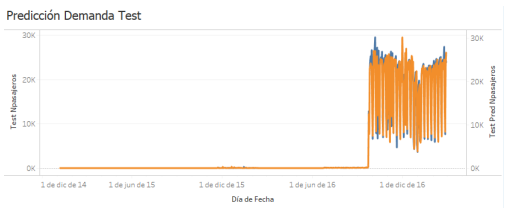


Figura 6-9.: Dashboard: Gráfica Predicción Training.

Hay que indicar que para cada una de las pantallas y hojas en el dashboard se han implementado una serie de filtros para que el usuario pueda realizar un buen filtrado de los datos que desea visualizar.

En el Anexo III podemos ver la representación completa del Dashboard implementado.

7. Conclusiones y Trabajo Futuro

7.1. Conclusiones

En este proyecto se ha llevado a cabo la implementación de un proyecto en el área del Data Science y Big Data para la mejora en uno de los transportes públicos más usados en Madrid. En lo referente a los objetivos propuestos en el primer capítulo hay que resaltar:

- **Conclusión Objetivo 1: Estudio de los datos de transporte público**
  En lo referente al estudio de los datos del sector transporte, hemos comprobado que lo principal para este tipo de proyectos es encontrar datos referentes al número de viajeros en el tiempo, tipología de las líneas, fuentes externas que aporten información relevante en el estudio, etc.

- **Conclusión Objetivo 2: Estudio de fuentes de datos externas al transporte público**
  El estudio de fuentes externas a los datos de la EMT, hay que indicar que tras analizar múltiples fuentes, se ha concluido que las principales fuentes externas a tener en cuenta en un proyecto de este tipo son fuentes que hacen referencia a la meteorología, tráfico, eventos de toda índole (principalmente los eventos deportivos) y el calendario laboral.

- **Conclusión Objetivo 3: Estudio de la ingesta de datos en Hadoop**
  Una de las partes importantes de este y otros proyectos similares es la ingesta de estos en plataformas Big Data. En nuestro proyecto, hemos realizado una ingesta lo más automatizada posible para que pueda ser exportable a otros proyectos similares, así como a la ingesta de nuevos datos.

- **Conclusión Objetivo 4: Análisis de los datos en el transporte público.**
  Una vez que tenemos los datos cargados en nuestro sistema Big Data (cluster Hadoop) hemos analizado dichos datos mediante una auditoría de estos que nos ha permitido ver ciertas correlaciones entre las líneas, horarios de alta demanda y variables externas, que pueden ser de gran importancia para la EMT y para el posterior modelo a realizar.

- **Conclusión Objetivo 5: Técnicas de segmentación y predicción de la demanda.**
  Mediante los algoritmos realizados para la segmentación y predicción de la demanda

se ha visto que a partir de los datos de los viajeros y de las fuentes externas podemos ayudar a la EMT a segmentar las líneas por afluencia de viajeros y darles una estimación del número de viajeros a futuro para que planifiquen mejor su demanda y horarios.

- **Conclusión Objetivo 6: Visualización de los resultados.**
  Para finalizar, otra de las partes importantes en este tipo de proyectos es la visualización de los resultados de una forma adecuada para ayudar a los administradores de la plataforma a tomar decisiones basadas en los datos.

7.2. Trabajo Futuro

En cuanto al trabajo futuro por realizar, se han tenido en cuenta los siguientes puntos que no se han realizado debido a que el tiempo de realización del proyecto es limitado o estaban fuera del alcance de nuestros objetivos:

- **Ingesta Automática:** uno de los puntos a mejorar es la ingesta de una forma automática de los datos a nuestra base de datos en Hadoop. Esto lo podríamos realizar mediante herramientas como Sqoop o directamente desde Spark.
- **Fuentes Externas:** añadir más fuentes externas que nos permitan definir mejor nuestro problema. Estas pueden ser el turismo, datos de google, etc.
- **Fase de Predicción:** realizar la fase de Deep Learning dentro del cluster y realizarlo para cada una de las líneas de forma automática. Para ello necesitaremos instalar todas las dependencias de las librerías utilizadas para esta fase.
- **Realizar más casos de uso:** realizar algún caso de uso más a parte de los dos ya realizados y que sean de gran importancia para la EMT.
- **Mejora de los modelos:** mejorar los modelos ya realizados teniendo en cuenta información más detallada sobre aspectos particulares de negocio.

Bibliografía

[1] R. Kitchin (2014). The real-time city? Big data and smart urbanism. *GeoJournal*, 79(1), 1–14.

[2] A. Wren y D.O. Wren (1995). A genetic algorithm for public transport driver scheduling. *Computers Operations Research*, 22(1), 101–10.

[3] B.M. Williams y L.A. Hoel (2003). Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *Journal of Transportation Engineering*, 129, 664–72.

[4] J.L. Toole, S. Colak, B. Sturt, L.P. Alexander, A. Evsukoff y M.C. González (2015). The path most traveled: Travel demand estimation using big data resources. *Transportation Research Part C: Emerging Technologies*, 58, Part B, 162–77.

[5] J. Odeck y A. Alkadi (2001). Evaluating efficiency in the Norwegian bus industry using data envelopment analysis. *Transportation*, 28(3), 211–32.

[6] J. Lin, E. Keogh, S. Lonardi y B. Chiu (2003). A symbolic representation of time series, with implications for streaming algorithms, en *DMKD '03 Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2-11.

[7] C. Edelbrock (1979). Mixture model tests of hierarchical clustering algorithms: the problem of classifying everybody. *Multivariate Behavioral Research*, 14(3), 367–84.

[8] T.W. Liao (2005). Clustering of time series data — a survey. *Pattern Recognition*, 38(11), 1857-74.

[9] T.-C. Fu (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1), 164-81.

[10] P. Esling y C. Agon (2012). Time-Series Data Mining. *ACM Computing Surveys*, 45(1), 12:1–12:34

[11] C. Chatfield (2000). *Time-Series Forecasting*. Chapman & Hall/CRC.

[12] C. Winston y V. Maheshri (2007). On the social desirability of urban rail transit systems. *Journal of Urban Economics*, 62(2), 362-82.

[13] J. Brownlee *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ Accedido por última vez: Agosto 2017.

[14] J. Brownlee *Long Short-Term Memory Networks With Python*. https://machinelearningmastery.com/lstms-with-python/ Accedido por última vez: Agosto 2017.

[15] E. Busseti, I. Osband y S. Wong (2012). Deep Learning for Time Series Modeling. *CS 229 Final Project Report*. http://cs229.stanford.edu/proj2012/BussetiOsbandWong-DeepLearningForTimeSeriesModeling.pdf. Accedido por última vez: Agosto 2017.

[16] http://sqoop.apache.org/ Accedido por última vez: Agosto 2017.

[17] J. McQueen (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1, 281–97.

[18] A.K. Jain, M.N. Murty y P.J. Flynn (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3), 264–323.

[19] T. Hastie, R. Tibshirani y J. Friedman (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer.

[20] L. Breiman, J.H. Friedman, R.A. Olshen y C.J. Stone (1984). *Classification and regression trees*, 45(1). Wadsworth Brooks/Cole Advanced Books Software.

[21] L. Breiman (2001). Random Forests. *Machine Learning*, 45(1), 5–32.

[22] J. Schmidhuber (2015). Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61, 85–117.

[23] Y. Bengio, A. Courville y P. Vincent (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798—1828.

[24] Y. LeCun, Y. Bengio y G. Hinton (2015). Deep learning. *Nature*, 521, 436—44.

[25] S. Hochreiter y J. Schmidhuber (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-80.

[26] F.A. Gers, J. Schmidhuber y F. Cummins (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10), 2451—71

[27] F. Chollet y otros (2015). Keras. GitHub, https://github.com/fchollet/keras. Accedido por última vez: Septiembre 2017.

A. Anexo I: Scripts y Código

Se han realizado una serie de scripts divididos en las siguientes secciones:

- **Scripts de Ingesta :** se han realizado una serie de scripts de ingesta mediante los cuales se puedan ingestar de una forma automática todos los datos en nuestra base de datos Hadoop tan solo con ejecutar un único ejecutable. Hay que tener en cuenta que se han realizado scripts para la ingesta desde el propio cluster y otros scripts para la ingesta desde un pc.
- **Scripts de Auditoría :** también se han creado unos notebooks para analizar el comportamiento de las lineas, así como la estructura de los datos, para posteriormente poder aplicar los modelos de una forma más correcta y adecuada. En dichos scripts podemos ver gráficas y resultados ante las preguntas que se han ido realizando a lo largo del desarrollo del proyecto y que por lo tanto son un buen punto de partida para la resolución de nuestros problemas.
- **Scripts de Analítica :** finalmente se han desarrollado los scripts y notebooks mediante los cuales podemos obtener la parte de la analítica de nuestro proyecto. Tenemos dos secciones una para la fase de clustering y otra para la fase de predicción.

Todos ellos los podemos encontrar en el siguiente repositorio de github:
https://github.com/CarlosRosado/SmartBus.git

B. Anexo II: Dendograma Clustering Jerárquico Laborables

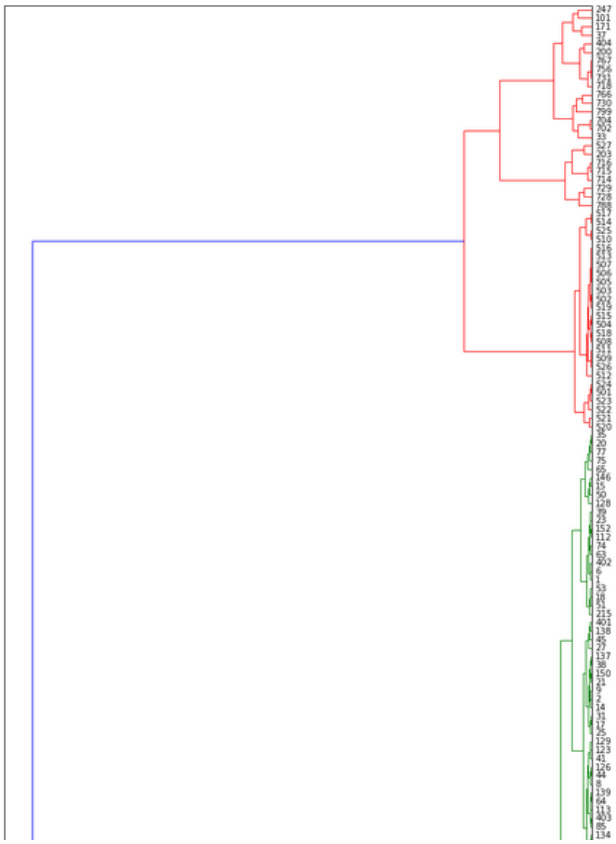


Figura B-1.: Dendograma Clustering Laborables (part1).

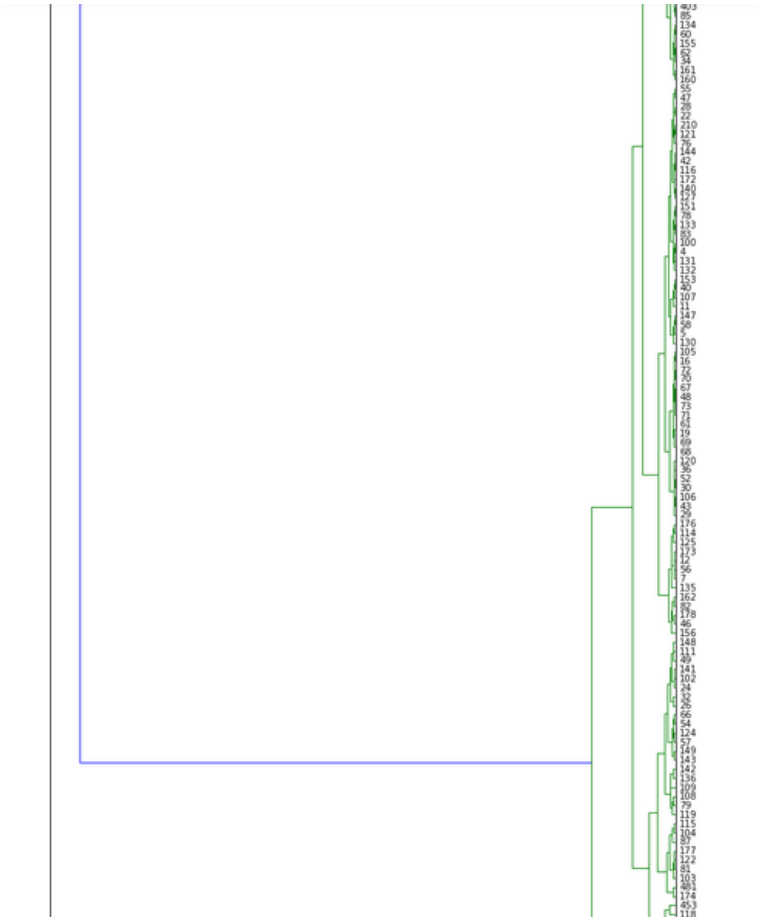


Figura B-2.: Dendograma Clustering Laborables (part2).

Figura B-3.: Dendograma Clustering Laborables (part3).

C. Anexo III: Pantallas Dashboard

Figura C-1.: Dashboard: Pantalla de Inicio

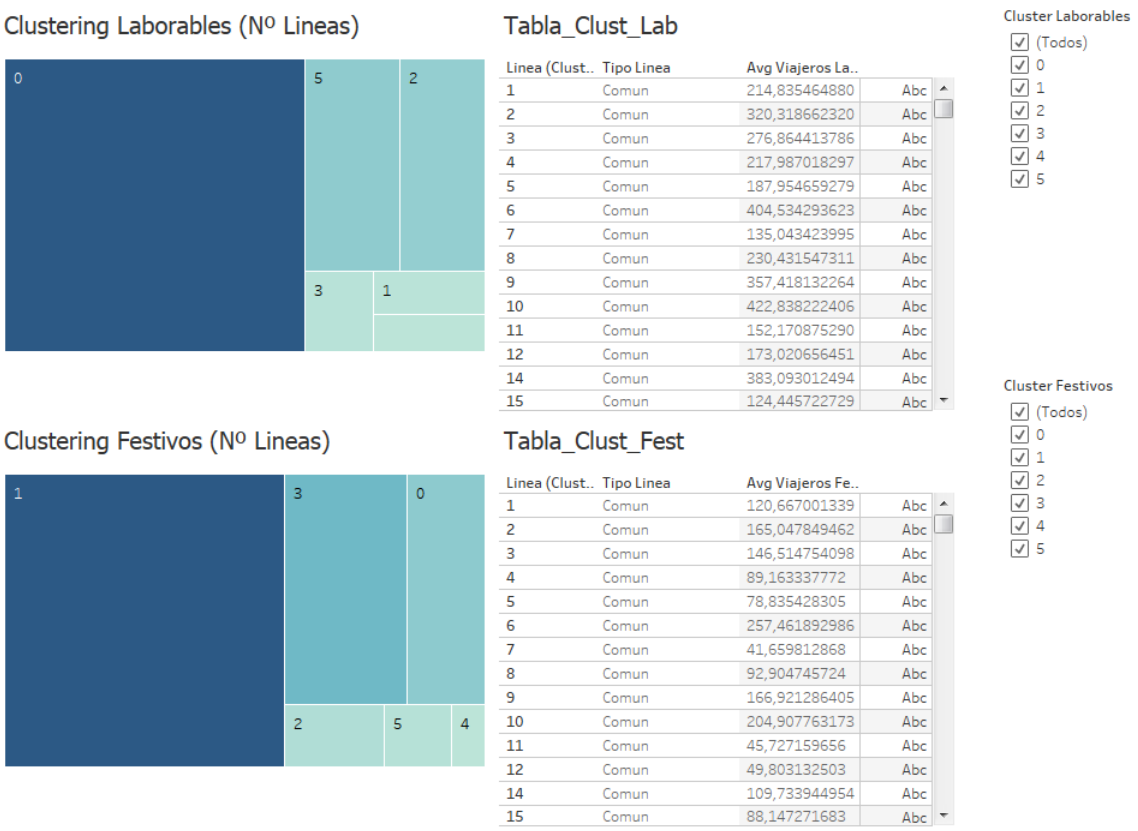


Figura C-2.: Dashboard: Pantalla de Clustering

Figura C-3.: Dashboard: Pantalla de Predicción de la Demanda